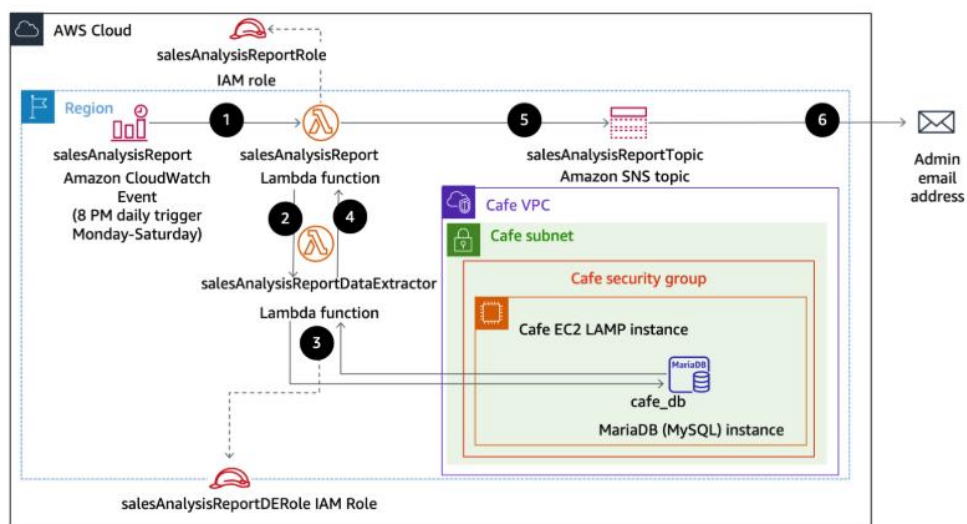


Activity - Working with AWS Lambda

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 <code>salesAnalysisReport</code> Lambda function at 8 PM every day Monday through Saturday.
2	The <code>salesAnalysisReport</code> Lambda function invokes another Lambda function, <code>salesAnalysisReportDataExtractor</code> , 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.
-

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:
 - a. **AmazonSNSFullAccess** provides full access to Amazon SNS resources.
 - b. **AmazonSSMReadOnlyAccess** provides read-only access to Systems Manager resources.
 - c. **AWSLambdaBasicRunRole** provides write permissions to CloudWatch logs (which are required by every Lambda function).
 - d. **AWSLambdaRole** gives a Lambda function the ability to invoke another Lambda function.

The salesAnalysisReport Lambda function that you create later in this lab uses the salesAnalysisReportRole role.

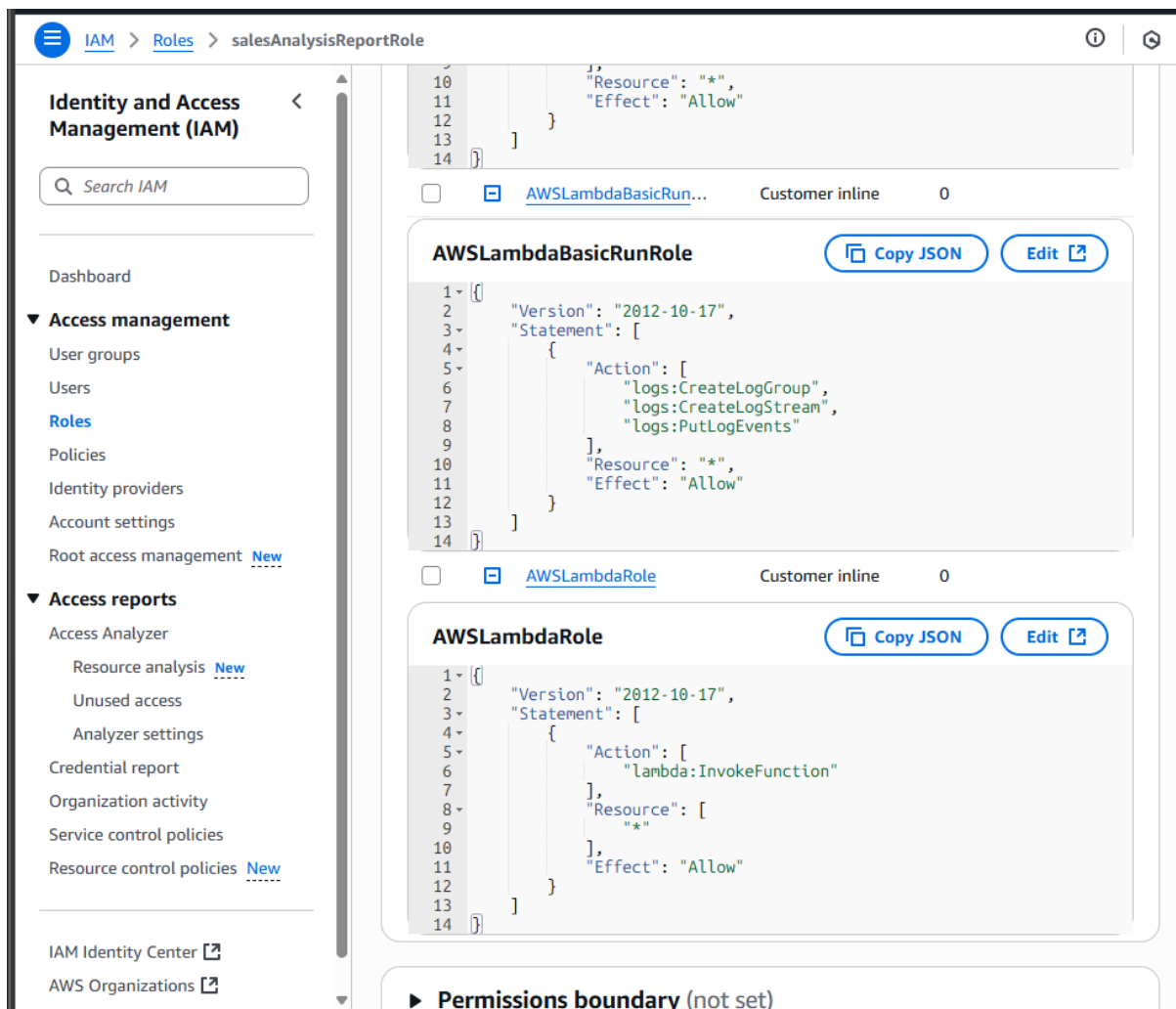
<

Q

[AWS Organizations](#)

>

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": {
7         "Service": "lambda
                        .amazonaws.com"
8       },
9       "Action": "sts:AssumeRole"
10    }
11  ]
12 }
```

Task 1.2: Observing the salesAnalysisReportDERole IAM role settings

11. Choose **Roles** again.
12. In the search box, enter `sa`
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:
 - a. **AWSLambdaBasicRunRole** provides write permissions to CloudWatch logs.
 - b. **AWSLambdaVPCElasticNetworkInterfaceRole** 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.

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings
- Root access management [New](#)

▼ Access reports

- Access Analyzer
- Resource analysis [New](#)
- Unused access
- Analyzer settings
- Credential report
- Organization activity
- Service control policies
- Resource control policies [New](#)

IAM Identity Center [↗](#)

AWS Organizations [↗](#)

salesAnalysisReportDERole [Info](#)

[Delete](#) [Edit](#)

Summary

Creation date
September 11, 2025, 16:47 (UTC+01:00)

ARN
[arn:aws:iam::387551059157:role/salesAnalysisReportDERole](#)

Last activity
-

Maximum session duration
1 hour

Permissions **Trust relationships** **Tags (1)** **Last Accessed** **Revoke sessions**

Trusted entities [Edit trust policy](#)

Entities that can assume this role under specified conditions.

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "Service": "lambda.amazonaws.com"  
8       },  
9       "Action": "sts:AssumeRole"  
10    }  
11  ]  
12 }
```

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings
- Root access management [New](#)

▼ Access reports

- Access Analyzer
- Resource analysis [New](#)
- Unused access
- Analyzer settings
- Credential report
- Organization activity
- Service control policies
- Resource control policies [New](#)

IAM Identity Center [↗](#)

AWS Organizations [↗](#)

salesAnalysisReportDERole

Policy name **type** **Attached entities**

<input type="checkbox"/> AWSLambdaBasicRunRole	Customer inline	0
<input type="checkbox"/> AWSLambdaVPCAccessRunRole	Customer inline	0

AWSLambdaBasicRunRole [Copy JSON](#) [Edit](#)

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Action": [  
6         "logs:CreateLogGroup",  
7         "logs:CreateLogStream",  
8         "logs:PutLogEvents"  
9       ],  
10      "Resource": "*",  
11      "Effect": "Allow"  
12    }  
13  ]  
14 }
```

AWSLambdaVPCAccessRunRole [Copy JSON](#) [Edit](#)

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Action": [  
6         "logs:CreateLogGroup",  
7         "logs:CreateLogStream",  
8         "logs:PutLogEvents",  
9         "ec2:CreateNetworkInterface",  
10        "ec2:DescribeNetworkInterfaces",  
11        "ec2:DeleteNetworkInterface"  
12      ],  
13      "Resource": "*",  
14      "Effect": "Allow"  
15    }  
16  ]  
17 }
```

► **Permissions boundary** (not set)

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:
 - a. For **Name**, enter pymysqlLibrary
 - b. For **Description**, enter PyMySQL library modules

- c. 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.
- d. 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 pymysqlLibrary.zip file used in this lab was packaged using the following folder structure:



For more information about layer paths, see [Including Library Dependencies in a Layer](#).

The screenshot shows the AWS Lambda console interface for a layer named 'pymysqlLibrary'. At the top, a green banner indicates 'Successfully created layer pymysqlLibrary version 1'. Below this, the layer name 'pymysqlLibrary' is displayed with buttons for 'Delete', 'Download', and 'Create version'. The 'Version details' section shows the following information:

Version details		
Version 1	Version ARN arn:aws:lambda:us-west-2:387551059157:layer:pymysqlLibrary:1	Description PyMySQL library modules
Created 1 second ago	License -	Compatible runtimes -
Compatible architectures -		

Below the details, there are tabs for 'Versions' and 'Functions using this version'. The 'All versions (1)' section shows a table with the following data:

Version	Version ARN	Description
1	arn:aws:lambda:us-west-2:387551059157:layer:pymysqlLibrary:1	PyMySQL library modules

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."

The screenshot displays the AWS Lambda console interface for the function `salesAnalysisReportDataExtractor`. At the top, a green notification banner states: "Successfully created the function salesAnalysisReportDataExtractor. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name is prominently displayed with buttons for **Throttle**, **Copy ARN**, and **Actions**. The **Function overview** section includes tabs for **Diagram** and **Template**, a visual representation of the function, and buttons for **Add trigger** and **Add destination**. To the right, details such as **Description**, **Last modified** (41 seconds ago), **Function ARN** (`arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReportDataExtractor`), and **Function URL** are shown. A navigation bar at the bottom allows switching between **Code**, **Test**, **Monitor**, **Configuration**, **Aliases**, and **Versions**. The **Code source** section is active, showing a preview of the `lambda_function.py` file within a Visual Studio Code-like editor interface.

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:
 - a. **Choose a layer:** Choose **Custom layers**.
 - b. **Custom layers:** Choose **pymysqlLibrary**.
 - c. **Version:** Choose **1**.
28. Choose **Add**.

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

The screenshot shows the AWS Lambda console interface. At the top, there's a breadcrumb navigation: **Lambda** > **Functions** > **salesAnalysisReportDataExtractor**. Below this, there are two green success messages: "Successfully updated the function salesAnalysisReportDataExtractor." and "Successfully created layer pymysqlLibrary version 1.".

The main section is titled **salesAnalysisReportDataExtractor**. It includes buttons for **Throttle**, **Copy ARN**, and **Actions**. Below this is the **Function overview** panel, which has tabs for **Diagram** and **Template**. The **Diagram** tab is active, showing a visual representation of the function with a **Layers** node containing **(1)** layer. To the right of the diagram, there's a **Description** section with fields for **Last modified** (1 second ago) and **Function ARN** (arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReportDataExtractor). There are also buttons for **Export to Infrastructure Composer** and **Download**.

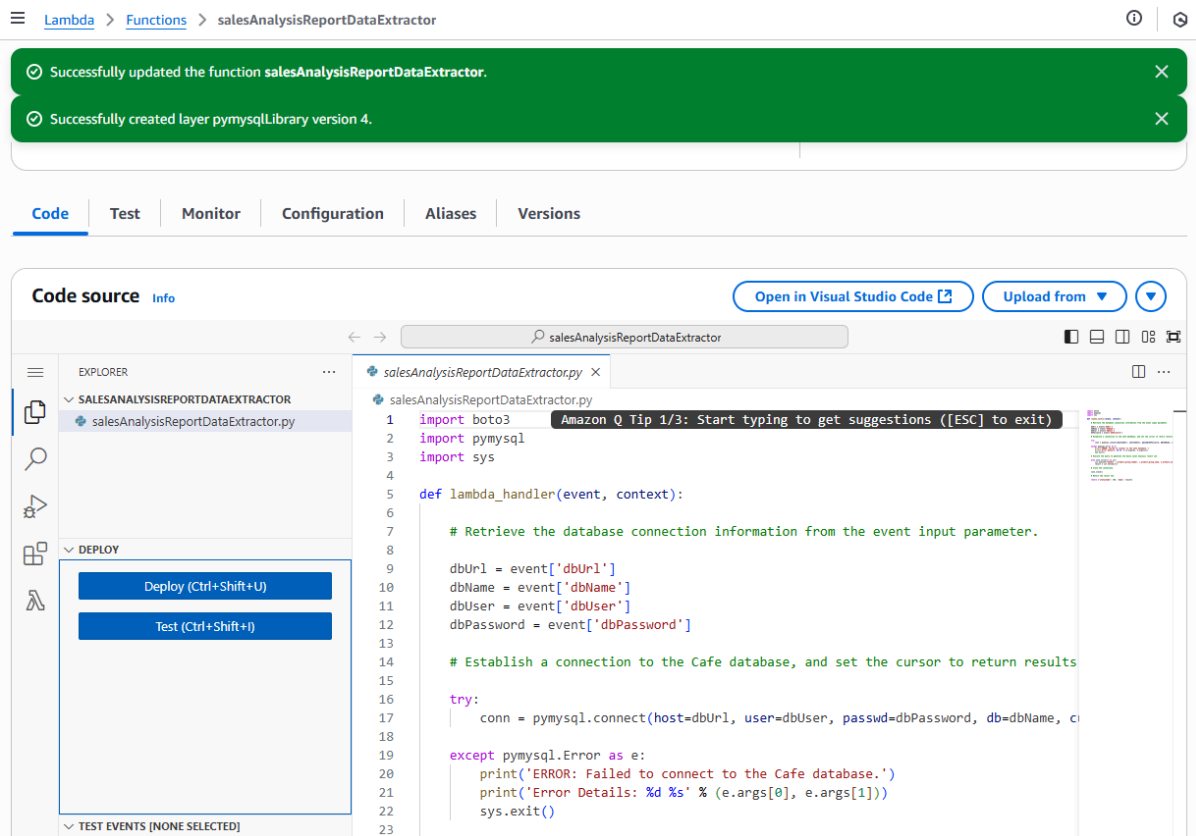
Below the **Function overview** panel is a horizontal navigation bar with tabs: **Code**, **Test**, **Monitor**, **Configuration**, **Aliases**, and **Versions**. The **Code** tab is selected, showing the **Code source** panel. This panel has buttons for **Open in Visual Studio Code** and **Upload from**. Below this, there's a preview of the code editor showing the **lambda_function.py** file.

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**.

The screenshot displays the AWS Lambda console interface for the function `salesAnalysisReportDataExtractor`. At the top, a green notification bar indicates successful updates to the function and the creation of a `pymysqlLibrary` layer. The **Function overview** panel shows the function's name, a diagram of its components, and buttons for adding triggers and destinations. The **Code source** panel is active, showing the `lambda_function.py` file in the Explorer. The **Configuration** tab is selected, and the **Runtime settings** panel is visible, showing the handler `salesAnalysisReportDataExtractor.lambda_handler`.

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**.

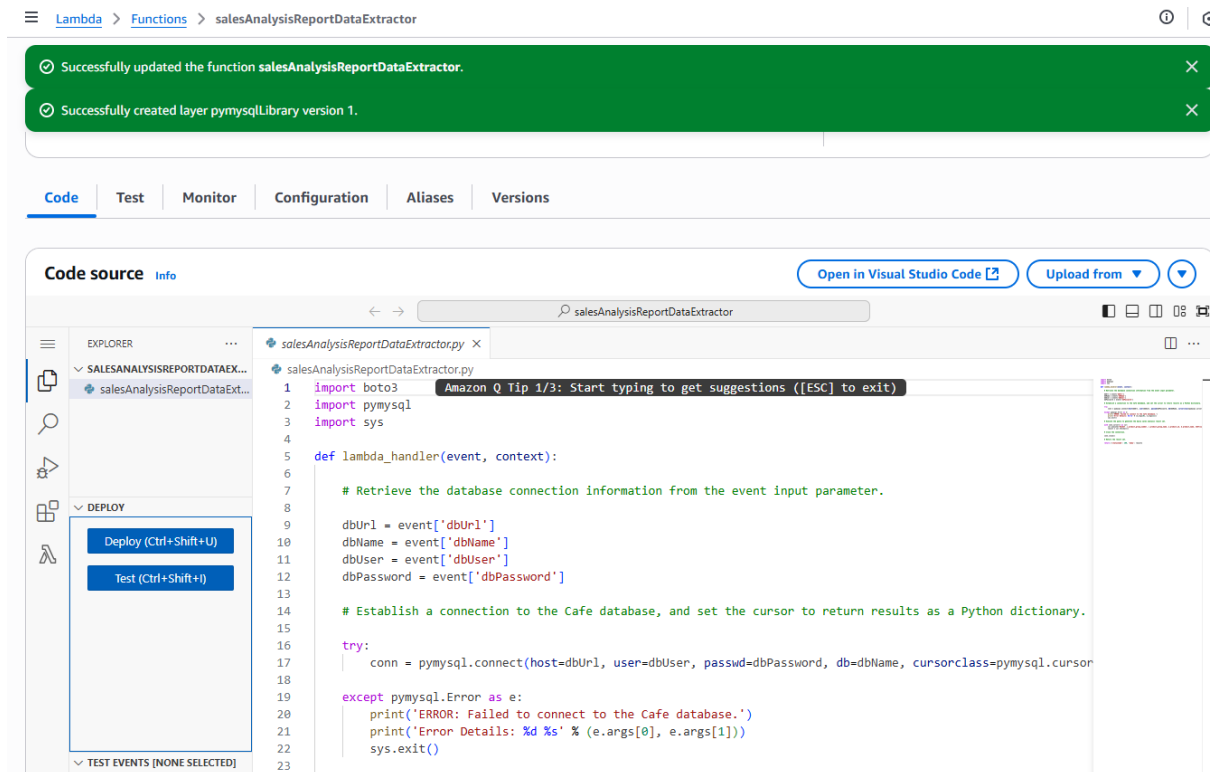


The Lambda function code is imported and displays in the **Code source** panel. If necessary, in the **Environment** navigation pane, double-click **salesAnalysisReportDataExtractor.py** to display the code.

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:
 - a. **VPC**: Choose the option with **Cafe VPC** as the **Name**.
 - b. **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.

- c. **Security groups**: Choose the option with **CafeSecurityGroup** as the **Name**.

Notice that the security group's inbound and outbound rules are automatically displayed following the field.

40. Choose **Save**.

The screenshot shows the AWS Lambda console interface for the function `salesAnalysisReportDataExtractor`. At the top, there are two status bars: a blue one indicating an update to the function and a green one indicating the successful creation of a `pymysqlLibrary` layer version 4. The function name `salesAnalysisReportDataExtractor` is displayed at the top right, along with buttons for `Throttle`, `Copy ARN`, and `Actions`. Below this, the `Function overview` tab is active, showing a diagram of the function, its layers (one `pymysqlLibrary` layer), and buttons to `Add trigger` and `Add destination`. To the right, the `Description` field is empty, and the `Last modified` time is 5 minutes ago. The `Function ARN` is `arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReportDataExtractor`, and the `Function URL` is also empty. Below the overview, the `Configuration` tab is selected, showing the `VPC` configuration (VPC ID: `vpc-098d1e8d87360a660`, Subnet: `subnet-0ffdc5363ccd74ba`), `Subnets` (Allow IPv6 traffic: false, Subnet: `subnet-0ffdc5363ccd74ba`), and `Security groups` (Security group ID: `sg-078858ecd72f14459`, Security group name: `CafeSecurityGroup-PCvPKIYI68Wg`). The `Security groups` field is highlighted with a blue border.

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:
 - a. `/cafe/dbUrl`

- b. /cafe/dbName
- c. /cafe/dbUser
- d. /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:

- a. For **Test event action**, select **Create new event**.
- b. For **Event name**, enter SARDETestEvent
- c. For **Template**, choose **hello-world**.
- d. In the **Event JSON** pane, replace the JSON object with the following JSON object:

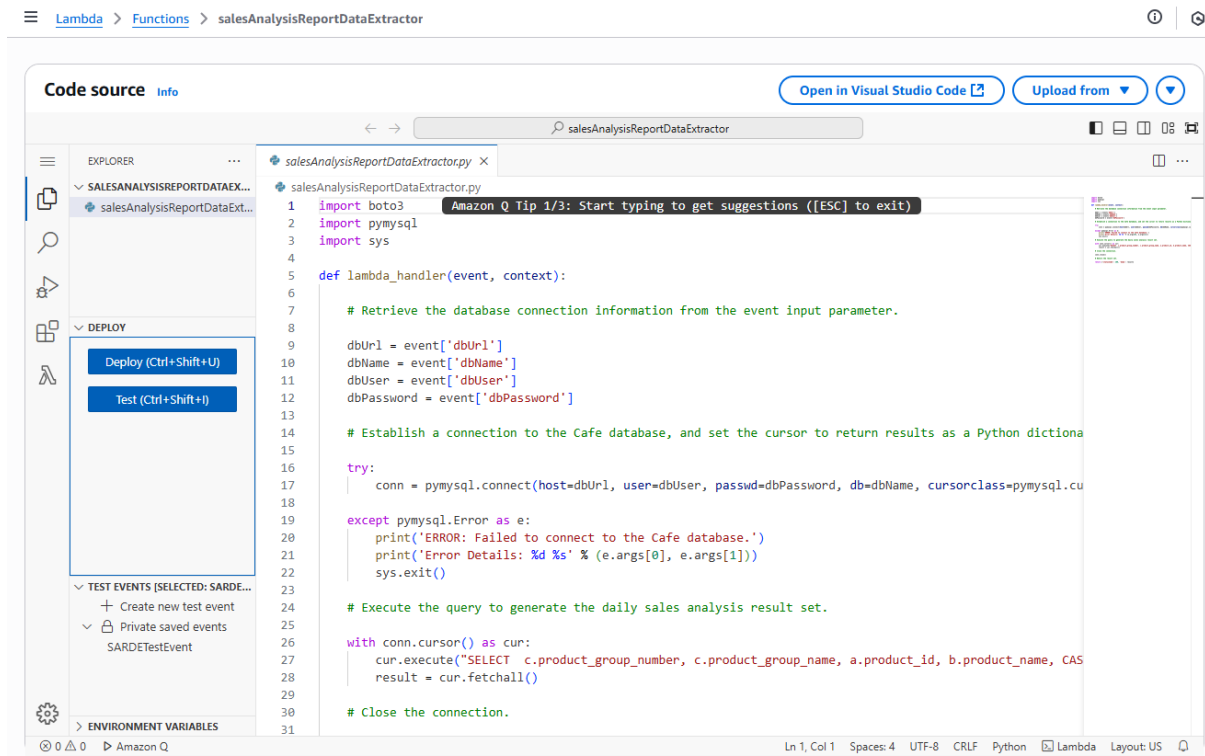
```
{  
  "dbUrl": "<value of /cafe/dbUrl parameter>",  "dbName": "<value of  
/cafe/dbName parameter>",  "dbUser": "<value of /cafe/dbUser  
parameter>",  "dbPassword": "<value of /cafe/dbPassword parameter>"}
```

- e. 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:

```
{
  "errorMessage": "2019-02-14T04:14:15.282Z ff0c3e8f-1985-44a3-8022-519f883c8412 Task timed out after 3.00 seconds"}
```

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.

What caused this error?

Lambda > Functions > salesAnalysisReportDataExtractor

salesAnalysisReportDataExtractor

Layers

(1)

+ Add trigger

+ Add destination

Last modified

9 minutes ago

Function ARN

arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReportDataExtractor

Function URL

Info

Code

Test

Monitor

Configuration

Aliases

Versions

Executing function: failed (logs)

Diagnose with Amazon Q

Test event Info

Delete

CloudWatch Logs Live Tail

Save

Test

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save.

Test event action

Create new event

Edit saved event

Event name

SARDETestEvent

Event JSON

Format JSON

The test event "SARDETestEvent" was successfully saved.

Executing function: failed (logs)

Diagnose with Amazon Q

Details

```
{  "errorMessage": "Unable to import module 'salesAnalysisReportDataExtractor': No module named 'pymysql'",  "errorType": "Runtime.ImportModuleError",  "requestId": "",  "stackTrace": []}
```

Summary

Code SHA-256	Execution time
IRqIWdxSEyWeJ36QlnarwO0FSsPmjKtXZ+jWsQU+S6o=	1 minute ago
Function version	Request ID
\$LATEST	0c4c8114-8f48-48ef-96b6-699ddf8626c3
Duration	Billed duration
241.59 ms	242 ms
Resources configured	Max memory used
128 MB	57 MB

Log output

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

[WARNING] 2025-09-22T13:48:18.378Z LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in rare cases, a Lambda runtime update can cause unexpected function behavior. For functions using managed runtimes, runtime updates can be triggered by a function change, or can be applied automatically. To determine if the runtime has been updated, check the runtime version in the INIT_START log entry. If this error correlates with a change in the runtime version, you may be able to mitigate this error by temporarily rolling back to the previous runtime version. For more information, see https://docs.aws.amazon.com/lambda/latest/dg/runtimes-update.html

[ERROR] Runtime.ImportModuleError: Unable to import module 'salesAnalysisReportDataExtractor': No module named 'pymysql'

Traceback (most recent call last):

INIT_REPORT Init Duration: 230.17 ms Phase: invoke Status: error Error Type: Runtime.ImportModuleError

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:

- a. 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.
 - b. By default, a MySQL database uses the MySQL protocol and listens on port number 3306 for client access.
 - c. 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.

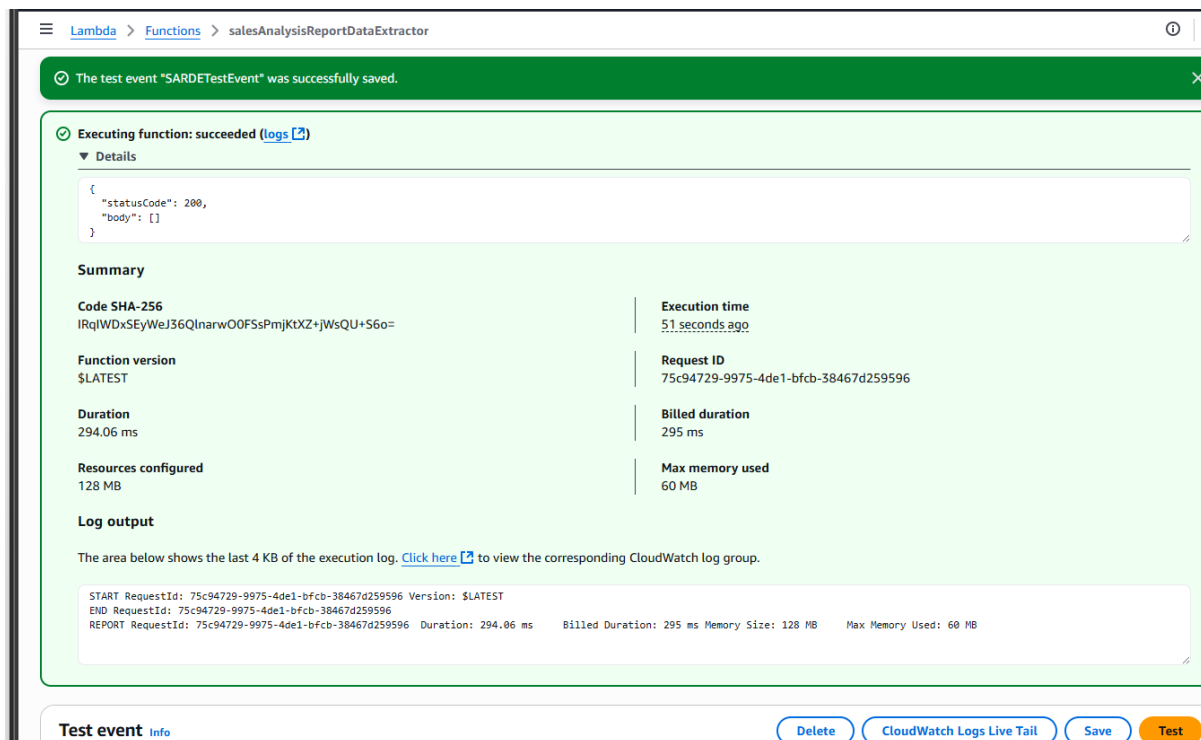
You should now see a green box showing the message “Execution result: succeeded (logs).” This message indicates that the function ran successfully.

51. Choose **Details** to expand it.

The function returned the following JSON object:

```
{
  "statusCode": 200,
  "body": []
}
```

The body field, which contains the report data that the function extracted, is empty because there is no order data in the database.



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.

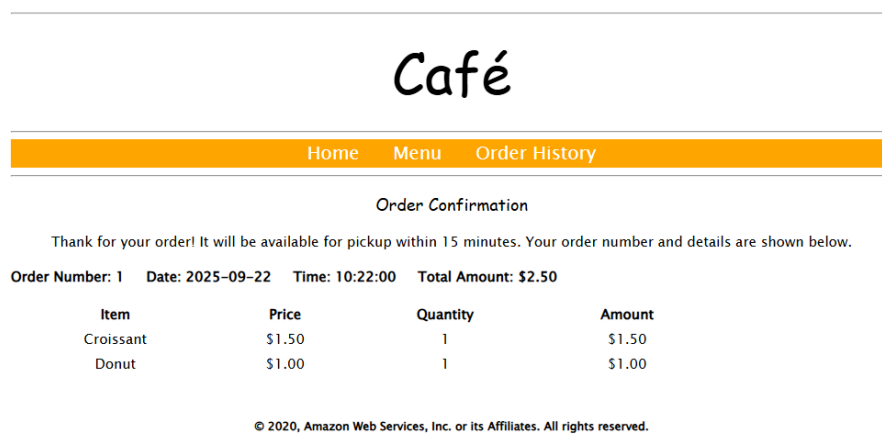
The URL for the website has the format <http://publicIP/cafe> where **publicIP** is the public IP address of the café EC2 instance. There are two ways to find the public IP address:

Option 1:

- a. On the AWS Management Console, choose **Services** > **Compute** > **EC2**.
- b. In the navigation pane, choose **Instances**.
- c. Choose **CafeInstance**.
- d. Copy the **Public IPv4 address** to a text editor.
- e. 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.
- f. Press Enter to load the café website.

Option 2:

- g. At the top of these instruction, choose Details, and then choose Show.
 - h. From the **Credentials** window, copy and paste the **CafePublicIP** to a text editor.
 - i. 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.
 - j. Press Enter to load the café website.
53. 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.

54. Go to the browser tab with the **salesAnalysisReportDataExtractor** function page.

55. Choose the **Test** tab, and choose **Test**.

The returned JSON object now contains product quantity information in the body field similar to the following:

```
{
  "statusCode": 200,
  "body": [
    {
      "product_group_name": "Pastries",
      "product_group_number": 1,
      "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    }    ]}

```

Congratulations! You have successfully created the salesAnalysisReportDataExtractor Lambda function.

The screenshot shows the AWS Lambda console interface. At the top, there are buttons for '+ Add trigger' and '+ Add destination'. Below these are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Test' tab is selected, showing a green banner that says 'Executing function: succeeded (logs [?])'. Underneath, the 'Details' section shows a JSON response:

```

{
  "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
    }
  ]
}

```

Below the details, a 'Summary' section provides key metrics:

Code SHA-256 IRqIWDxSEyWeJ3GQInarwO0FsPmjKtXZ+jWsQU+56o=	Execution time 11 seconds ago
Function version \$LATEST	Request ID 063643a8-48fc-471e-a159-19028ac2f33a
Duration 161.37 ms	Billed duration 162 ms
Resources configured	Max memory used

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:

- a. **Type:** Choose **Standard**.
- b. **Name:** Enter `salesAnalysisReportTopic`
- c. **Display name:** Enter `SARTopic`

59. Choose **Create topic**.

The screenshot shows the Amazon SNS console interface. On the left is a navigation pane with 'Amazon SNS' selected, and sub-links for 'Dashboard', 'Topics', 'Subscriptions', and 'Mobile'. The main content area shows the 'salesAnalysisReportTopic' details. At the top, there are two notification banners: a blue one about 'New Feature' and a green one stating 'Topic salesAnalysisReportTopic created successfully'. Below these are buttons for 'Edit', 'Delete', and 'Publish message'. The 'Details' section contains a table with the following information:

Details	
Name salesAnalysisReportTopic	Display name SARTopic
ARN arn:aws:sns:us-west-2:387551059157:salesAnalysisReportTopic	Topic owner 387551059157
Type Standard	

Below the details is a tabbed interface with 'Subscriptions' selected. It shows 'Subscriptions (0)' with buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription'. A search bar is present. Below the search bar is a table header with columns: ID, Endpoint, Status, and Protocol. The table is currently empty, displaying the message 'No subscriptions found. You don't have any subscriptions to this topic.' with a 'Create subscription' button below it.

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:

- a. **Protocol:** Choose **Email**.
- b. **Endpoint:** Enter an email address that you can access.

62. Choose **Create subscription**.

The screenshot shows the Amazon SNS console interface. The breadcrumb navigation at the top reads: Amazon SNS > Topics > salesAnalysisReportTopic > Subscription: 86b293d8-6fa1-4fd1-b4a2-6acc83168950. On the left sidebar, under 'Amazon SNS', the 'Subscriptions' link is highlighted. Below it, under 'Mobile', are 'Push notifications' and 'Text messaging (SMS)'. The main content area features a blue banner with a 'New Feature' announcement and a green success message: 'Subscription to salesAnalysisReportTopic created successfully. The ARN of the subscription is arn:aws:sns:us-west-2:387551059157:salesAnalysisReportTopic:86b293d8-6fa1-4fd1-b4a2-6acc83168950.' Below the messages, the subscription title 'Subscription: 86b293d8-6fa1-4fd1-b4a2-6acc83168950' is displayed with 'Edit' and 'Delete' buttons. A 'Details' section follows, containing a table with the following information:

Details	Status	Protocol
ARN arn:aws:sns:us-west-2:387551059157:salesAnalysisReportTopic:86b293d8-6fa1-4fd1-b4a2-6acc83168950	⌚ Pending confirmation	EMAIL
Endpoint barbieru_mariana@yahoo.com		
Topic salesAnalysisReportTopic		
Subscription Principal arn:aws:iam::387551059157:role/voclabs		

Below the details, there are tabs for 'Subscription filter policy' (selected) and 'Redrive policy (dead-letter queue)'. The 'Subscription filter policy' section states: 'This policy filters the messages that a subscriber receives. No filter policy configured for this subscription. To apply a filter policy, edit this subscription.' with an 'Edit' button.

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!"



Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

arn:aws:sns:us-west-2:387551059157:salesAnalysisReportTopic:86b293d8-6fa1-4fd1-b4a2-6acc83168950

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

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:

```
aws configure
```

70. At the prompts, enter the following information:
 - a. **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.
 - b. **AWS Secret Access Key:** From the same **Credentials** window, copy the **SecretKey** value, paste it into the terminal window, and press Enter.
 - c. **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.
 - d. **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:

```
cd activity-files
```

1s

Note: Before you create the function, you must retrieve the ARN of the salesAnalysisReportRole IAM role. You specify it in the following steps.

```
#  
~\_##### Amazon Linux 2  
~~\_#####\  
~~\_####| AL2 End of Life is 2026-06-30.  
~~\_#/ \  
~~ V~' '->  
~~~~  
~~.-. A newer version of Amazon Linux is available!  
~/_/_/_/  
_/m/'_/_/_/ Amazon Linux 2023, GA and supported until 2028-03-15.  
https://aws.amazon.com/linux/amazon-linux-2023/  
  
[ec2-user@ip-10-200-0-149 ~]$ aws configure  
AWS Access Key ID [None]: AKIAVUO6X3TKWNLN4S6Q  
AWS Secret Access Key [None]: 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  
Default region name [None]: us-west-2  
Default output format [None]: json  
[ec2-user@ip-10-200-0-149 ~]$ cd activity-files  
[ec2-user@ip-10-200-0-149 activity-files]$ ls  
salesAnalysisReport-v2.zip  
[ec2-user@ip-10-200-0-149 activity-files]$
```

72. To find the ARN of an IAM role, open the IAM management console, and choose **Roles**.
73. In the search box, enter **salesAnalysisReportRole**, and choose the role name. The **Summary** page includes the **ARN**.
74. Copy and paste the ARN to a text editor document.
75. 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.

```
~~~~~
~~~~~
/m/'-_/ A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

ec2-user@ip-10-200-0-149 ~]$ aws configure
WS Access Key ID [None]: AKIAVUO6X3TKWNLN4S6Q
WS Secret Access Key [None]: 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
default region name [None]: us-west-2
default output format [None]: json
ec2-user@ip-10-200-0-149 ~]$ cd activity-files
ec2-user@ip-10-200-0-149 activity-files]$ ls
salesAnalysisReport-v2.zip
ec2-user@ip-10-200-0-149 activity-files]$ aws lambda create-function \
--function-name salesAnalysisReport \
--runtime python3.9 \
--zip-file fileb://salesAnalysisReport-v2.zip \
--handler salesAnalysisReport.lambda_handler \
--region us-west-2 \
--role arn:aws:iam::387551059157:role/salesAnalysisReportRole

{
  "FunctionName": "salesAnalysisReport",
  "LastModified": "2025-09-22T14:34:11.280+0000",
  "RevisionId": "1a4b4fef-b1cd-4b0f-98dc-108849f4f341",
  "MemorySize": 128,
  "State": "Pending",
  "Version": "$LATEST",
  "Role": "arn:aws:iam::387551059157:role/salesAnalysisReportRole",
  "Timeout": 3,
  "StateReason": "The function is being created.",
  "Runtime": "python3.9",
  "StateReasonCode": "Creating",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "CodeSha256": "FOQaNphpQr/canEnzctygYFVreHKiABxYNh8X8lOpnE=",
  "Description": "",
  "CodeSize": 1643,
  "FunctionArn": "arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReport",
  "Handler": "salesAnalysisReport.lambda_handler"
}
```

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

The screenshot shows the AWS Lambda console interface for the function `salesAnalysisReport`. At the top, a green notification bar states "Successfully updated the function salesAnalysisReport." The function overview section shows the function name, a diagram icon, and a "Layers" section with 0 layers. The "Configuration" tab is selected, and the "Environment variables" section is expanded, showing a table with one variable:

Key	Value
topicARN	arn:aws:iam::387551059157:role/salesAnalysisReportRole

The "Environment variables" section also includes a search bar and a note: "The environment variables below are encrypted at rest with the default Lambda service key."

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 SARDETestEvent
- For **Template**, choose **hello-world**.

The function does not require any input parameters. Leave the default JSON lines as is.

84. Choose **Save**.

85. Choose **Test**.

The screenshot shows the AWS Lambda console interface for the function `salesAnalysisReport`. At the top, there are two green success messages: "Successfully updated the function salesAnalysisReport." and "The test event 'SARDETestEvent' was successfully saved." Below these, the function's details are shown, including the Lambda icon, the name `salesAnalysisReport`, and a button to "Add trigger". To the right, there are buttons for "Add destination" and "Test". The "Test" tab is selected, showing the "Test event" configuration. The "Test event action" is set to "Create new event", and the "Event name" is `SARDETestEvent`. The "Test" button is highlighted in orange. The "Function ARN" is `arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReport`.

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**.

86. Choose **Details** to expand it.

The screenshot shows the AWS Lambda console for the function `salesAnalysisReport`. At the top, there are two green notification banners: "Successfully updated the function salesAnalysisReport." and "The test event 'SARDETestEvent' was successfully saved." Below these, the configuration section shows the function is triggered by an `event` and has no layers. The right sidebar displays the function's ARN and URL. The `Test` tab is selected, showing a successful execution of the function. The details section shows the response body: `{"statusCode": 200, "body": "\"Sale Analysis Report sent.\""}`. The summary section provides metadata such as the code SHA-256, execution time (25 seconds ago), function version (\$LATEST), request ID, and billed duration.

Notifications: 0 0 2 0 0

Function ARN: `arn:aws:lambda:us-west-2:387551059157:function:salesAnalysisReport`

Function URL: [Info](#)

Executing function: succeeded ([logs](#))

Details

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

Summary

Code SHA-256 FOQaNphpQr/canEnzctygYFVreHKiABxYNh8X8lOpnE=	Execution time 25 seconds ago
Function version \$LATEST	Request ID df4d77f6-711d-4d5a-9db9-c0f693a32547
Duration	Billed duration

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:

```
Sales Analysis Report
Date: 2019-02-15

Product Group: Pastries


Item Name      Quantity
*****
Croissant      1
Donut          2
Chocolate Chip Cookie  7
Muffin         4
Strawberry Blueberry Tart  9
Strawberry Tart  6

Product Group: Drinks

Item Name      Quantity
*****
Coffee         7
Hot Chocolate  10
Latte          9
```

Daily Sales Analysis Report

barbieru_marian.../Inbox

 **SARTopic** <no-reply@sns.amazonaws.com>
To: barbieru_mariana@yahoo.com

22 Sept at 15:48 ● ☆
[Print](#) [Raw message](#)

Sales Analysis Report
Date: 2025-09-22

Product Group: Pastries

Item Name	Quantity
*****	*****
Croissant	1
Donut	1

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:387551059157:salesAnalysisReportTopic:86b293d8-6fa1-4fd1-b4a2-6acc83168950&Endpoint=barbieru_mariana@yahoo.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

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:

- a. In the **Trigger configuration** pane, from the dropdown list, choose **EventBridge (CloudWatch Events)**.
- b. For **Rule**, choose **Create a new rule**.
- c. For **Rule name**, enter salesAnalysisReportDailyTrigger
- d. For **Rule description**, enter Initiates report generation on a daily basis
- e. For **Rule type**, choose **Schedule expression**.
- f. 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:
 - i. 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:

- ii. 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 *)
```

- iii. 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.

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

91. Choose **Add**.

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

The screenshot shows the AWS Lambda console interface for the function `salesAnalysisReport`. At the top, a green notification bar states: "The test event 'SARDETestEvent' was successfully saved." Below this, a dark notification bar shows "Notifications" with counts: 0 errors, 0 warnings, 2 info, and 0 debugs. The function name `salesAnalysisReport` is displayed with buttons for `Throttle`, `Copy ARN`, and `Actions`. A green notification bar below the name states: "The trigger salesAnalysisReportDailyTrigger was successfully added to function salesAnalysisReport. The function is now receiving events from the trigger." The main section is titled **Function overview** and includes tabs for `Diagram` and `Template`. The `Diagram` tab shows a visual representation of the function with its layers and an EventBridge (CloudWatch Events) trigger. A button `+ Add destination` is visible. To the right, a `Description` panel shows the function's last modified time (4 minutes ago) and its ARN: `arn:aws:lambda:us-west-2:387551059:157:function:salesAnalysisReport`. Below the overview, there are tabs for `Code`, `Test`, `Monitor`, `Configuration` (selected), `Aliases`, and `Versions`. The `Configuration` tab shows the **General configuration** section with an `Edit` button.

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.