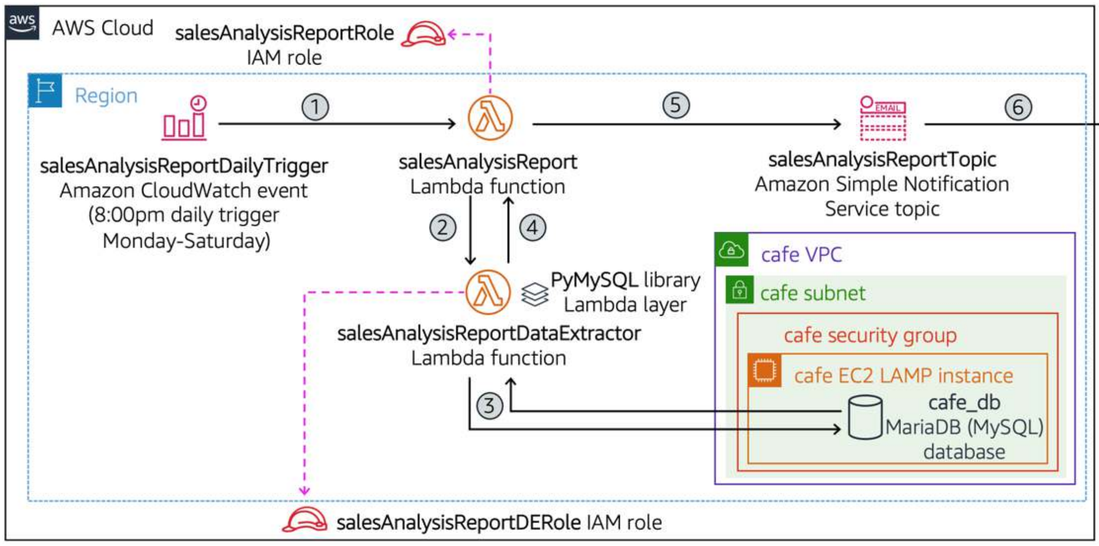
Activity - Working with AWS Lambda

Activity Overview

In this activity, you deploy and configure an AWS Lambda-based serverless computing solution. The Lambda function will generate a *Sales Analysis Report*, by pulling data from a database and emailing the results out on a daily basis. The database connection information is stored in the AWS Systems Manager (SSM) Parameter Store. The database itself runs on an Amazon EC2 LAMP instance.

The diagram here shows the architecture of the Sales Analysis Report solution and illustrates the order in which actions occur.



The function steps shown in the diagram above are as follows:

- [1] An Amazon CloudWatch event triggers the *salesAnalysisReport* Lambda function at 8:00pm 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 (SNS) topic.
- [6] The *salesAnalysisReportTopic* sends the message by email to Frank.

During this activity, 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.

Activity Objectives

After completing this activity, you will be able to:

- **Recognize** necessary IAM policy permissions to enable a Lambda function to other AWS resources.
- **Create** a Lambda layer to satisfy an external library dependency.
- **Create** a Lambda function.
- **Deploy** and test a Lambda function that is triggered based on a schedule and that invokes another function.
- **Use** CloudWatch logs to troubleshoot any issues running a Lambda function.

Business Case Relevance

A New Business Requirement for Café – Daily Sales Analysis Report

Frank is pleased that the website tracks orders placed online. Every day after closing, he now prints out the Order History page and figures out the quantity of each product sold that day. With this information, he can project how many of each product he needs to prepare the next day to better match supply with demand.

However, he has found this manual task to be time consuming. He wished the application would perform the sales analysis automatically and send him a report by email at the end of each day. Could this enhancement be implemented?

Nikhil and Sofîa consulted with the Mateo from the AWS Team who recommended using AWS Lambda.

It would be a cost-effective way to meet the business requirement. Take on the role of Nikhil and Sofîa, and deploy a Lambda-based solution to generate and distribute the sales analysis report.

Activity Steps

Duration: This activity requires approximately **60 minutes** to complete.

Accessing the AWS Management Console

1. At the top of these instructions, click Start Lab to launch your lab.

   A Start Lab panel opens displaying the lab status.

2. Wait until you see the message "**Lab status: ready**", then click the **X** to close the Start Lab panel.
3. At the top of these instructions, click AWS

   This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

   Tip: If a new browser tab does not open, there will typically be a banner or icon at the top of your browser indicating that your browser is preventing the site from opening pop-up windows. Click on the banner or icon and choose "Allow pop ups."

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time, to make it easier to follow the lab steps.

   Leave this browser tab open. You will return to it later in this activity.

Task 1: Observe the IAM Role settings

In this activity you will create two Lambda functions. Each of those functions will require permissions to access the AWS resources with which they interact. In this task, you analyze the IAM roles and the permissions they will grant to the salesAnalysisReport and salesAnalysisReportDataExtractor Lambda functions that you will create later.

Task 1.1: Observe the salesAnalysisReport IAM Role settings

5. In the AWS Management Console select **Services** > **Security, Identity, & Compliance** > **IAM**.
6. Click **Roles**.
7. In the Roles detail page, click into the search box and type `sales`.
8. In the filtered results, click on the **salesAnalysisReportRole** hyperlink.
9. Click on the **Trust relationships** tab and notice that *lambda.amazonaws.com* is listed as a trusted entity, meaning that the Lambda service can use this role.

10. Click on the **Permissions** tab and notice the four policies assigned to this role. Expand each one in turn (by clicking on the triangle icon) and analyze the permissions each policy grants:

    - **AmazonSNSFullAccess**: provides full access to Amazon SNS resources.
    - **AmazonSSMReadOnlyAccess**: provides read-only access to AWS SystemsManager resources.
    - **AWSLambdaBasicRunRole**: provides write permissions to CloudWatch logs (required by every Lambda function).
    - **AWSLambdaRole**: allows a Lambda function to invoke another Lambda function.

    This role will be used by the sales analysis report lambda function you will create later in this exercise.

Task 1.2: Observe the salesAnalysisReportDERole IAM Role settings

11. Click on **Roles** again, then click into the search box and type `sales` again.
12. In the filtered results, click on **salesAnalysisReportDERole**.
13. Click on the **Trust relationships** tab and notice that lambda.amazonaws.com is listed as a trusted entity.
14. Click on 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 VPC.

    This role will be used by the salesAnalysisReportDataExtractor Lambda function you will create next.

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

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

Start by downloading two required files.

15. Download the activity files required by this task to your local machine by clicking on each of the links below:

    pymysql-v3.zip

    salesAnalysisReportDataExtractor-v3.zip

    Note: *salesAnalysisReportDataExtractor* 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 pymysql-0.9.3.zip.

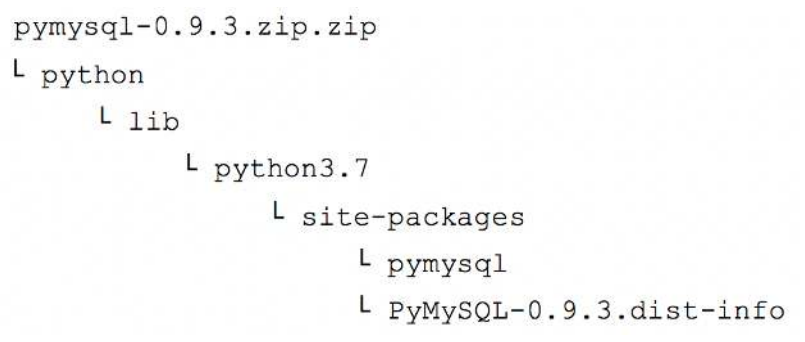
Task 2.1: Create a Lambda Layer

In the next steps, you will 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 by eliminating the need to include the code in each function's deployment package.

16. In the AWS Management Console, select **Services** > **Compute** > **Lambda**.
17. If the navigation panel is closed, click on the collapsed menu icon (three horizontal lines icon)—just underneath the AWS logo—to open the *AWS Lambda* panel.
18. Click **Layers**.
19. Click **Create layer**.
20. Configure the layer settings as follows:
    - **Name**: `pymysqlLibrary`
    - **Description**: `PyMySQL 0.9.3 library modules`
    - **Code entry type**: Upload a .zip file
    - Click **Upload**, navigate to the folder where you downloaded **pymysql-0.9.3.zip** and open it.
    - **Compatible runtimes**: Choose **Python 3.9**.
21. Click **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 to reuse conform to a specific folder structure. The *pymysqlLibary.zip* file used in this activity was packaged using the following folder structure:



```
pymysql-0.9.3.zip.zip
└ python
      └ lib
            └ python3.7
                  └ site-packages
                        └ pymysql
                        └ PyMySQL-0.9.3.dist-info
```

    For more information on Layer paths, see here.

Task 2.2: Create the data extractor Lambda function

22. In the "breadcrumb" navigation trail at the top, click **Lambda** to open the Lambda Dashboard page.
23. Click **Create function** and configure as follows:
    - **Author from scratch**: (selected)
    - **Function name**: `salesAnalysisReportDataExtractor`

    - **Runtime**: Python 3.9
    - Expand **Change default execution role**, then configure the following:
        - **Execution role**: Use an existing role
        - **Existing role:**: salesAnalysisReportDERole
24. Click **Create function**.

    A new page opens with the message: *Successfully created the function salesAnalysisReportDataExtractor.*

Task 2.3: Add the Lambda Layer to the function

25. In the *Function overview* panel, click **Layers**.
26. In the *Layers* panel at the bottom of the page, click **Add a layer**.
27. In the *Add layer* page, configure as follows:
    - **Choose a layer**: Select the *Custom layers* card
    - **Custom layers**: pymysqlLibrary
    - **Version**: 1
28. Click **Add**.

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

Task 2.4: Import the data extractor Lambda function code

29. Make sure you are on the **Lambda > Functions > salesAnalysisReportDataExtractor** page
30. Scroll down to the *Runtime settings* panel, and choose **Edit**.
31. In the *Handler* field, change the value to `salesAnalysisReportDataExtractor.lambda_handler`.
32. Click **Save**.
33. In the *Code source* panel, click **Upload from**.
34. Select **.zip file**.
35. Click **Upload**, then navigate to and select the **salesAnalysisReportDataExtractor-v2.zip** you downloaded earlier.
36. Click **Save**.

    The Lambda function code is imported and should display in the *Code source* panel. If necessary, double-click **salesAnalysisReportDataExtractor.py** in the left navigation pane.
37. Take a moment to 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: Configure network settings for the function

The final step before you can test the function is to configure its network settings. As shown in the architecture diagram at the start of this activity, 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. Select the **Configuration** tab, and click **VPC**.
39. Click **Edit** and configure as follows:
    - **VPC**: CafeVPC
    - **Subnets**: Cafe Public Subnet 1

      TIP: You can ignore the warning that recommends choosing at least 2 subnets to run in high availability mode because it is not applicable to the function.
    - **Security groups**: CafeSecurityGroup

      Notice that the security group’s inbound and outbound rules are automatically displayed below the field.
40. Click **Save**.

Task 3: Test the data extractor Lambda function

Task 3.1: Launch a test of the Lambda function

You are now ready to unit test the salesAnalysisReportDataExtractor function. In order to invoke it, you will need to supply values for the Café database connection parameters. Recall that these are stored in the AWS Systems Manager Parameter Store.

41. In a new browser tab, open the AWS Management Console and select **Services** > **Management & Governance** > **Systems Manager**.
42. Click on **Parameter Store** and record the values for the following parameter names (for example, copy and paste them in a text editor document):
    - /cafe/dbUrl
    - /cafe/dbName
    - /cafe/dbUser
    - /cafe/dbPassword
43. Return to the **Lambda Management Console** browser tab, and click the **Test** tab.
44. Configure the *Test event* panel as follows:
    - Make sure the **Create new event** radio button is selected.
    - **Event name**: `SARDETestEvent`

- **Event template**: Hello World (displays as *hello-world*)
- Replace the JSON object in the editor pane below Event name with the following:

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

Important: Be sure to substitute the appropriate parameter values that you recorded in a previous step and to enclose them in double-quotes.

45. Click **Save**.

46. Click **Test**.

    After a few moments, a red box displays at the top of the page showing the message *“Execution result: failed(logs)”*.

Task 3.2: Troubleshooting the data extractor Lambda function

47. In the error box, expand the **Details** section. Notice the error object returned after the function ran:

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

The message indicates that the function timed out after 3 seconds.

Look at the *Log output* section at the bottom of the error box. It shows lines starting with the following keywords:

- START - indicates that the function started running.
- END - indicates that function finished running.
- REPORT - provides a summary of the performance and resource utilization statistics related to when the function ran.

The last line of the log output also shows the time out error message. So, what caused this error?

Task 3.3: Lambda function troubleshooting challenge

48. Try to troubleshoot and correct this issue.

    Here are a few hints to help you find the solution:

    - One of the first things that this function does is to connect to the MySQL database running in a separate EC2 instance. It will wait a certain amount of time to establish a successful

connection, after which, if unsuccessful, the function will time out.
    - By default, a MySQL database uses the MySQL protocol and listens on port number 3306 for client access.
    - Select the **Configuration** tab again, and click **VPC**. Notice the *Inbound rules* for the security group used by the EC2 instance running the database. Is the database port number (3306) listed? Note that you can click the security group link to allow you to edit and add an inbound rule to it.

49. Once you have corrected the problem, return to the Lambda function's console and click **Test** again in the *Test* tab.

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

50. Expand the **Details** section.

    The function returned the following JSON object:

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

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

Task 3.4: Place an order and test again

51. In a new browser tab, access the Café website and place some orders to populate data in the database.

    As a reminder, the URL to open the website has the format http://publicIP/cafe where publicIP is the public IP address of the cafe EC2 instance. You can find the public IP in the EC2 service page in the AWS Management Console, or you can find it by clicking Show in the Details drop down menu above these instructions.

52. Now that there is order data in the database, test the function again. Click **Test**.

    Notice that the returned object now contains product quantity information in the body field, similar to the following:

    ```
    {
      "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
       }
      ]
  }
```

Congratulations! You have successfully created the salesAnalysisReportDataExtractor Lambda function.

Task 4: Configure notifications

In this task you will create a Simple Notification Service (SNS) topic and then subscribe an email address to the topic.

Task 4.1: Create a Simple Notification Service (SNS) topic

In this next task, you create the SNS topic where the Sales Analysis Report is published and subscribe Frank's email address to it. The topic is responsible for delivering any message it receives to all of its subscribers. You perform this task using the Amazon SNS Console.

53. Select **Services** > **Application Integration** > **Simple Notification Service**.
54. In the panel on the left, click **Topics** and then on the right side of the screen, click **Create topic**.

    Note: If the Topics link is not visible, click the three horizontal lines icon on the left and then click **Topics**.

    The *Create topic* page opens.
55. Configure the topic as follows:
    - **Type**: Standard
    - **Name**: `salesAnalysisReportTopic`
    - **Display name**: `SARTopic`
56. Click **Create topic**.
57. Record the value of the Topic *ARN* field (for example, copy it into a text document).

    You will need to specify it later when you configure the next Lambda function.

Task 4.2: Subscribe to the SNS topic

58. Click **Create subscription** and configure as shown:
    - **Protocol**: Email

    - **Endpoint**: (type in an email address that you can access)

    Note: For the purposes of this activity, you are going to pretend that you are Frank in order to receive the generated Sales Analysis Report.

59. Click **Create subscription**.

    The subscription is created and has a status of *Pending confirmation*.

60. Check the inbox for the email address that you provided.

    You should see an email from SARTopic with the subject AWS Notification - Subscription Confirmation.

61. Open the email and click **Confirm subscription**.

    A new browser tab should open and display a page with the message: Subscription confirmed!

Task 5: Create the sales analysis report Lambda function

Next, you will 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 the Systems Manager 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.

In this task, you will open an SSH session to the CLI Host instance running in your AWS account that already has the AWS CLI installed as well as the the Python code needed to create the next Lambda function. You will then then run an AWS CLI command to create the Lambda function. Finally, you will unit test the new function using the Lambda Management Console.

Windows users should follow Task 5.1. Both macOS and Linux users should follow Task 5.2.

macOS/Linux users—click here for login instructions

Task 5.1: Windows SSH

These instructions are for Windows users only.

If you are using macOS or Linux, skip to the next section.

62. Read through the three bullet points in this step before you start to complete the actions, because you will not be able see these instructions when the Details panel is open.

    - Click on the Details drop down menu above these instructions you are currently reading, and then click Show. A Credentials window will open.
    - Click on the **Download PPK** button and save the **labsuser.ppk** file. Typically your browser will save it to the Downloads directory.

  - Then exit the Details panel by clicking on the **X**.

63. Download needed software.
    - You will use **PuTTY** to SSH to Amazon EC2 instances. If you do not have PuTTY installed on your computer, download it here.
64. Open **putty.exe**.
65. Configure PuTTY to not timeout:
    - Click **Connection**.
    - Set **Seconds between keepalives** to `30`.

    This allows you to keep the PuTTY session open for a longer period of time.
66. Configure your PuTTY session:
    - Click **Session**.
    - **Host Name (or IP address):** Copy and paste the **IPv4 Public IP address** for the CLI Host instance. To find it, return to the EC2 Console and click on **Instances**. Check the box next to the CLI Host instance you want to connect to and in the *Details* tab copy the **Public IPv4 address** value.
    - Back in PuTTy, in the **Connection** list, expand ⊞ **SSH**.
    - Click **Auth** (don't expand it).
    - Click **Browse**.
    - Browse to and select the **labsuser.ppk** file that you downloaded.
    - Click **Open** to select it.
    - Click **Open**.
67. Click **Yes**, to trust the host and connect to it.
68. When prompted **login as**, enter: `ec2-user`

    This will connect you to the EC2 instance.

    Windows Users: Click here to skip ahead to the next task.

Task 5.2: macOS/Linux SSH

These instructions are for Mac/Linux users only. If you are a Windows user, skip ahead to the next task.

69. Read through the three bullet points in this step before you start to complete the actions, because you will not be able see these instructions when the Details panel is open.
    - Click on the Details drop down menu above these instructions you are currently reading, and then click Show. A Credentials window will open.
    - Click on the **Download PEM** button and save the **labsuser.pem** file.
    - Then exit the Details panel by clicking on the **X**.
70. Open a terminal window, and change directory `cd` to the directory where the labsuser.pem file was downloaded.

    For example, run this command, if it was saved to your Downloads directory:

```
cd ~/Downloads
```

71. Change the permissions on the key to be read only, by running this command:

```
chmod 400 labsuser.pem
```

72. Return to the AWS Management Console, and in the EC2 service, click on **Instances**. Check the box next to the CLI Host instance you want to connect to.
73. In the *Details* tab, copy the **Public IPv4 address** value.
74. Return to the terminal window and run this command (replace **<public-ip>** with the actual public IP address you copied):

```
ssh -i labsuser.pem ec2-user@<public-ip>
```

75. Type `yes` when prompted to allow a first connection to this remote SSH server.

    Because you are using a key pair for authentication, you will not be prompted for a password.

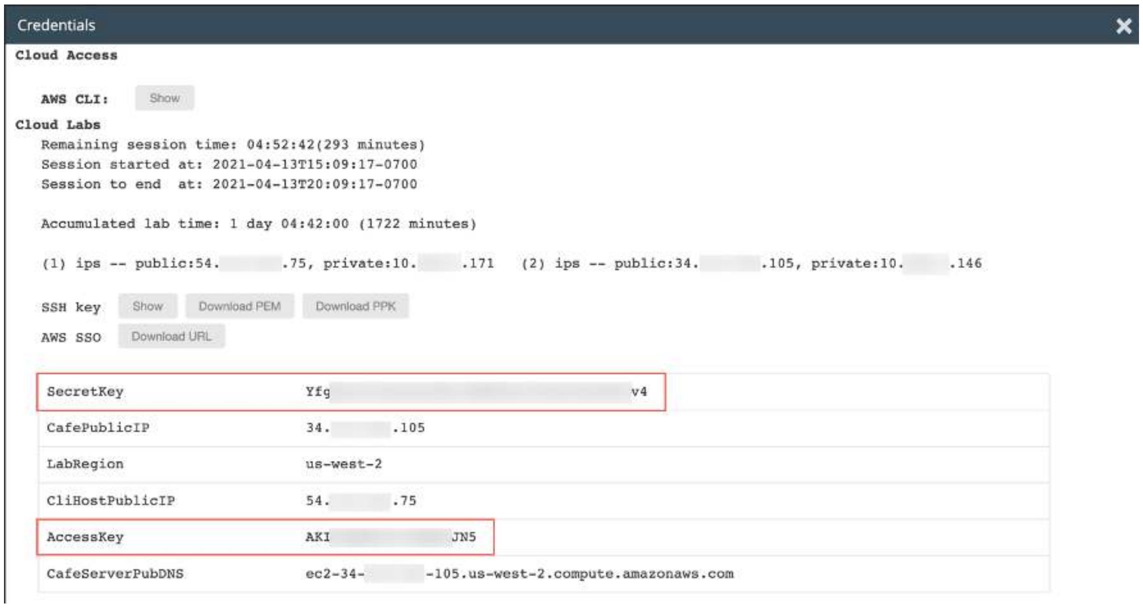
Task 5.3: Configure the AWS CLI

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

76. Update the AWS CLI software with the credentials.

```
aws configure
```

77. At the prompts, enter the following information:
    - **AWS Access Key ID**: Click on the Details drop down menu above these instructions, and then click Show. Copy the **AccessKey** value and paste it into the terminal window.



Credentials

Cloud Access

AWS CLI: Show

Cloud Labs

```
Remaining session time: 04:52:42(293 minutes)
Session started at: 2021-04-13T15:09:17-0700
Session to end  at: 2021-04-13T20:09:17-0700

Accumulated lab time: 1 day 04:42:00 (1722 minutes)

(1) ips -- public:54.[redacted].75, private:10.[redacted].171   (2) ips -- public:34.[redacted].105, private:10.[redacted].146
```

SSH key Show Download PEM Download PPK

AWS SSO Download URL

SecretKey	Yfg[redacted]v4
CafePublicIP	34.[redacted].105
LabRegion	us-west-2
CliHostPublicIP	54.[redacted].75
AccessKey	AKI[redacted]JN5
CafeServerPubDNS	ec2-34-[redacted]-105.us-west-2.compute.amazonaws.com

- **AWS Secret Access Key**: Copy and paste the **SecretKey** value from the same Credentials screen.
- **Default region name**: Specify the region code (for example `us-west-2`) of the region where you created the previous Lambda function. You can find this code by expanding the region name in the top menu bar of the Lambda Management Console.
- **Default output format**: `json`

Task 5.4: Create the salesAnalysisReport Lambda function using the AWS CLI

78. Verify that the file containing the code for the *salesAnalysisReport* Lambda function is already on the CLI Host by running the following commands:

```
cd activity-files
ls
```

79. Before you create the function, retrieve the ARN of the *salesAnalysisReportRole* IAM role. You will need to specify it in the next step.

    Tip: To find the ARN of an IAM role, open the IAM Management Console, click **Roles** and select the role name. The Role ARN appears on the *Summary* page.

80. Use the AWS Lambda *create-function* command to create the Lambda function and configure it to use the salesAnalysisReportRole IAM role.

    To do this, at the command prompt, paste in the following command. Replace **<salesAnalysisReportRoleARN>** with the actual value of the salesAnalysisReportRole ARN, which you copied a moment ago and replace **<region>** with the region code (for example `us-west-2`) of the region where you created the previous Lambda function. You can find this code by expanding the region name in the top menu bar of the Lambda Management Console.

```
aws lambda create-function \
--function-name salesAnalysisReport \
--runtime python3.7 \
--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 can now proceed to complete its configuration and unit test it.

Task 5.5: Configure the salesAnalysisReport Lambda function

81. Open the Lambda Management Console.
82. Click **Functions**, then click **salesAnalysisReport**.

    The detail page for the function is opened.

83. Take a moment to review the details in the *Funtion overview* and *Code source* panels of 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 will need to define that variable in the *Environment variables* panel.

84. Select the **Configuration** tab and click **Environment variables**.
85. Click **Edit > Add environment variable** and configure the fields as follows:
    - **Key**: `topicARN`
    - **Value**: (paste in the ARN value of the salesAnalysisReportTopic you copied earlier)
86. Click **Save**.

Task 5.6: Test the salesAnalysisReport Lambda function

You are now ready to test the function.

87. Select the **Test** tab and configure the test event as follows:
88. Configure as follows:
    - **Creat new event**: (selected)
    - **Template**: Hello World (displays as *hello-world*)
    - **Event name**: `SARTestEvent`

    The function does not require any input parameters (just leave the default JSON lines there).

89. Click **Save**.
90. Click **Test**.

    You should see a green box showing the message "Execution result: succeeded(logs)".

    Tip: If you get a *time out* error, click the **Test** button again. Sometimes, when you first run a function, it takes a little longer to initialize, and the AWS Lambda default timeout value (3 seconds) is exceeded. Usually, you can just run it again, and the error will go away. Alternatively, you can increase the timeout value in the *General configuration* panel.

91. Expand the **Details** section.

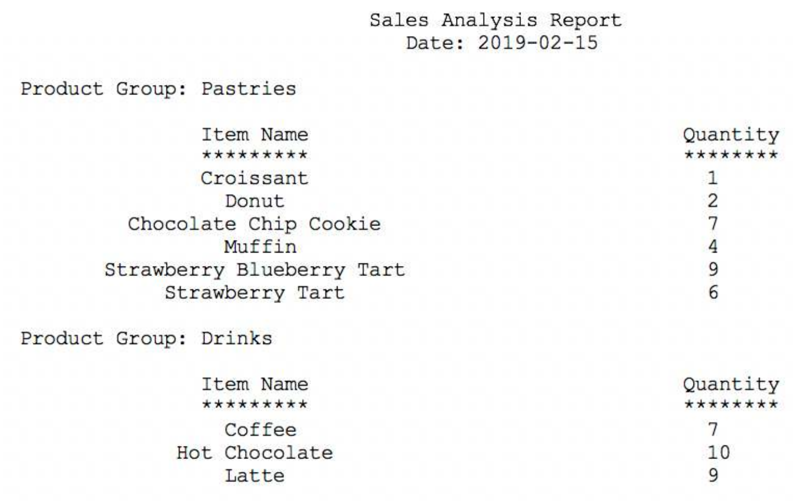
    The function should have returned the following JSON object:

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

92. 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 one shown here, depending on the orders you placed on the 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
```

93. Feel free to 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 5.7: Add a trigger to the salesAnalysisReport Lambda function

To complete the implementation of the *salesAnalysisReport*, configure the report to be triggered every day, Monday through Saturday, at 8:00pm. To do so, you use a CloudWatch event as the trigger mechanism.

94. In the *Function overview* panel, click **Add trigger**. The *Add trigger* panel is displayed.
95. In the *Add trigger* panel:

    Select **EventBridge (CloudWatch Events)**.

    - **Rule**: Create a new rule
    - **Rule name**: `salesAnalysisReportDailyTrigger`
    - **Rule description**: `Triggers report generation on a daily basis`
    - **Rule type**: Schedule expression

- **Schedule expression**: specify the schedule that you desire using a Cron expression.

  The general syntax of a Cron expression requires six fields separated by white space 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.

  For our testing purposes, type an expression that schedules the trigger 5 minutes from the current time. For example:

  - If you are in **London** (UTC time zone), and the current time is 11:30am, type the following expression:

    `cron(35 11 ? * MON-SAT *)`

  - If you are in **New York** (UTC time zone - 5), and the current time is 11:30am, type the following expression:

    `cron(35 16 ? * MON-SAT *)`

  This schedules the event to be triggered at 11:35am every day Monday through Saturday.

  Tip: For more information on the syntax of schedule expressions for rules, see https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html.

  You can navigate to **Google** and enter `UTC time` to get the right time

96. Click **Add**.

    The new trigger is created and displayed in the CloudWatch Events panel.

97. **Challenge question**:

    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 Friday. Assume the you are in the UTC time zone.

98. 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. This one was triggered by the CloudWatch event at the time that you specified in the Cron expression.

Update from Café

Nikhil and Sofîa arrived at the Cafe the next morning after deploying the Lambda serverless solution. As soon as Frank saw them walk in the door, he excitedly pulled out his cell phone, and showed them the email he had received with the order details in it.

He explained how helpful this new reporting capability was. Now he won't have to login every day to check the orders database details and manually calculate the orders placed. Instead, he just checks his email and he already knows how much of each item he will need to prepare.

Secretly, Frank hopes that the word doesn't get out about Nikhil and Sofîa's cutting edge technical skills and that recruiters for competing companies don't start approaching Nikhil and Sofîa to offer them higher paying jobs! He is sure they will be able to create even more helpful business enhancements to the Cafe cloud deployment in the future, if only he can get them to stay. Maybe he should offer them a raise or at least some free dessert! He is thrilled about what they have accomplished.

Activity Complete

Congratulations! You have completed the activity.

99. Click End Lab at the top of this page and then click **Yes** to confirm that you want to end the activity.

    A panel will appear, indicating that "DELETE has been initiated... You may close this message box now."

100. Click the **X** in the top right corner to close the panel.