Using Data Encryption in AWS

**SPL-TF-300-SIDTEN-10 - Version 1.0.11**

© 2023 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab Overview**

AWS offers numerous services that help protect data at rest and in transit. In this lab, you’ll explore a real-world situation and gain practical experience with multiple services. You’ll learn about Server Side Encryption (SSE) for Amazon S3, AWS Key Management Service (KMS), and Amazon DynamoDB Encryption Client. You’ll optionally use the AWS Encryption SDK to encrypt and decrypt data programmatically.

TOPICS COVERED

By the end of this lab, you will be able to:

* Create a KMS Key for encryption management
* Enable S3 Server-side Encryption
* Configure Attribute-Based Access Control
* Use the DynamoDB Encryption Client to encrypt database data in DynamoDB
* Encrypt data programmatically with the AWS Encryption SDK

PREREQUISITES

This lab requires:

* Google **Chrome** browser. The lab requires the students to use only Chrome as the internet browser because for Fleet Manager RDP, only **Chrome** browser supports bidirectional copying and pasting between RDP sessions and your local machine.
* Access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).
* Be familiar with the AWS console.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.
* **CAUTION:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

 You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Create a KMS Key for encryption management**

You are an administrator at Infinidash Healthcare - a new health services startup company - and you’ve been tasked with improving security in the organization by applying encryption technologies. Your first task is to create a KMS Key - an essential tool in allowing encryption of a variety of resources, including S3 storage objects, DynamoDB database tables, and more.

1. At the top of the AWS Management Console, in the search bar, search for and choose

KMS

.

1. Select **Create a key** .
2. Select Key type:

Symmetric

.

1. Under Advanced options, verify the following:

* Key material origin:

KMS

* Regionality:

Single-Region key

1. Select **Next** .
2. Specify the following settings:

* Alias:

ImageAccessKey

* Description:

This key is used to encrypt and decrypt customer medical imagery: MRI, XRay, etc.

 You may need to scroll up to see the Alias field.

1. Select **Add tag** .
2. Specify the following settings, which are CASE SENSITIVE

* Tag key:

ImageType

* Tag value:

MRI

Note: **do not** select **Add tag** a second time. If you accidentally add a second tag row, use the **Delete tag** button to remove it.

1. Select **Next** .

In the next few steps, you grant permissions to administer and use this new KMS Key. Administration permission will be attached to your Lab User Id. Usage permissions will be attached to the IAM Role that has already been configured for you in the AWS Identity and Access Management service and attached to the EC2 instance.

1. In **Step 3: Define key administrative permissions**, under **Key administrators**, in the search box with the  icon, type

user

.

1. Select  the user or role you’re signed into the Console with.

This user is displayed at the top of the page, to the right of the region.

1. Select **Next** .
2. In **Step 4: Define key usage permissions**, under **Key users**, search  for

view

 and select the checkbox next to

viewerInstanceRoleForMedicalImages

.

1. Select **Next** .
2. Select **Finish** .
3. In the list of **Customer managed keys**, select the link named

ImageAccessKey

.

1. In the **General configuration** area, copy the ARN to the clipboard using the  button. You will need this value in the next task.
2. Scroll down and verify that the user or role you are signed into the Console with, appears in the list of **Key administrators**, and that

viewerInstanceRoleForMedicalImages

 appears in the list of **Key users**.

1. Select the Key rotation tab.
2. Select the checkbox to

Automatically rotate this KMS key every year

.

1. Select **Save** .

**Congratulations!** You have successfully created a KMS key allowing encryption of a variety of resources, including S3 storage objects, DynamoDB database tables, and more.

**Task 2: Enable S3 Server-side Encryption**

In this second task, you enhance the security of your corporate storage data by configuring encryption for an S3 data storage bucket. **Note**: this change will not change the encryption status of the objects that are *already* in the bucket, but will ensure that *new* objects are stored in an encrypted state. Your KMS Key from Task 1 will give S3 the cryptographic material to secure new objects.

1. At the top of the AWS Management Console, in the search bar, search for and choose

S3

.

1. Select the blue bucket name link that starts with

corp-image-store-

 and note the current list of five objects in the bucket.

Infinidash Healthcare’s annual kids’ coloring contest appears to be underway. These images are not encrypted, but the incoming batch of XRay data certainly should be!

1. Scroll up and select the **Properties** tab.
2. In the **Default encryption** section, select the **Edit** button.
3. For **Encryption type**, select **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** .
4. Select

Enter AWS KMS key ARN

 under **AWS KMS key**.

1. Paste the copied ARN into the **AWS KMS key ARN** box.
2. Verify **Bucket Key** is set to

Enable

.

1. Choose **Save changes** .

**Expected service output:**

**Successfully edited default encryption. Objects uploaded, modified, or copied…**

**Congratulations!** Encryption is configured for the S3 bucket.

In the next task, use a tool to load new XRay images into the bucket, and observe the effects of our encryption settings. You are going to do that from an EC2 instance running the Windows Server operating system. This instance has permissions to access the corporate S3 bucket. You adjust these permissions over the course of this lab to configure secure access to corporate data.

**Task 3: Add new objects to the secured bucket**

In this task, you add objects to the S3 bucket.

RDP USING AWS FLEET MANAGER

In this task, you will connect to your Amazon EC2 instance using AWS Systems Manager Fleet Manager.

 Please make sure that you are using the **Chrome** as the Internet browser because for Fleet Manager RDP, only **Chrome** browser supports bidirectional copying and pasting between RDP sessions and your local machine.

 If you are unable to use **RDP** with **Fleet Manager**, you can also [connect to your windows instance using a Remote Desktop client](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-TF-300-SIDTEN-1%3A1.0.11-684b658d/en-US#connect_rdp).

**To connect to your Amazon EC2 instance using RDP with Fleet Manager**

1. To the left of the instructions you are currently reading, choose  **Download PEM**.
2. Save the file to the directory of your choice.
3. Get the **ViewerInstanceSessionRDP** values from the left side of these instructions.
4. Open **Google Chrome** on your local computer, and access **ViewerInstanceSessionRDP** from the browser.
5. For preferred **Authentication type** choose  Key pair.
6. For **Key pair content**, choose the following option  Browse your local machine to select the key pair file.
7. Select **Browse** to upload the **PEM** key from your local directory that is associated with your instance.
8. Select **Connect** .

 You will be prompted with a Networks pop-up window asking: **Do you want to allow your PC to be discoverable by other PCs and devices on this network?** Choose **No**.

 Fleet Manager RDP connections have a maximum session duration of 60 minutes. When that duration is reached, Fleet Manager disconnects the session. If you run into any issues while interacting with the Fleet Manager RDP, then choose **Actions** drop-down list, and then select **Renew session** to restart the duration timer.

* A Remote Desktop connection to a Windows Server desktop appears after several seconds. This server is used for loading XRay images from a vendor, and viewing them.

1. Launch the Image Transfer tool using the **Transfer Images** desktop shortcut.
2. Follow the prompts to transfer XRay images for a new patient into the S3 bucket. You will need the name of the **ImageStorageBucket** that appears in the AWS CONSOLE INFORMATION area to the left of these instructions.
3. When the ImageTransfer tool says: **Image transfer complete. Please close this window.**, close the Windows PowerShell window that contains that tool, returning to the Windows desktop.

The medical images vendor has transferred XRays to us. Now let’s confirm they have been stored using your KMS encryption key.

1. At the top of the AWS Management Console, in the search bar, search for and choose

S3

.

1. Select the blue bucket name link that starts with **corp-image-store-** and verify that there are now ten objects in the bucket. Use the refresh button  if necessary.
2. Select the blue link for one of the objects with a name starting with **Image-**.
3. Choose the **Properties** tab.
4. Scroll down to the **Server-side encryption settings** section and verify the following settings:

* Encryption type: AWS Key Management Service key (SSE-KMS)
* Encryption key ARN: (ARN from your ImagesKey created earlier)
* Bucket Key: Enabled

The XRay image objects are stored in an encrypted form. (The coloring contest images are not; feel free to confirm that for yourself if you wish.)

Note that encrypted and unencrypted objects can coexist in the same S3 bucket. The bucket-level encryption setting specifies the default behavior of *new* objects, but does not mandate encryption for all contents of the bucket.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Launch the Image Viewer tool using the **View Images** desktop shortcut. You will need the name of the **ImageStorageBucket** that appears in the AWS CONSOLE INFORMATION area to the left of these instructions.
3. Follow the prompts to view one or more of the **ColoringContest** images in the S3 bucket. Verify that this task succeeds.
4. Follow the prompts to view one or more of the **Image1.jpg**, **Image2.jpg**, etc. images in the S3 bucket. Note that attempting to view these files currently results in an **Access Denied** error. This is **expected** behavior.
5. When the Image Viewer tool says: “Display Another (Y/N)”, type

N

 followed by the Enter key, close the Windows PowerShell window that contains this tool, returning to the Windows desktop.

**Congratulations!** You have uploaded objects into the S3 bucket and encrypted them using your KMS encryption key.

**Task 4: Configure Attribute-Based Access Control**

You have successfully verified secure storage of sensitive customer data in S3 using KMS encryption. The user of the Windows instance has permission to access the kids’ unencrypted drawings, but not the encrypted XRay images. In this task, you reconfigure the settings to allow a medical technician to access those XRays.

1. At the top of the AWS Management Console, in the search bar, search for and choose

EC2

.

1. Select the check box for the instance named **Image Viewer EC2 Instance**.
2. Select the **Actions** drop-down menu, then select **Security** and then **Modify IAM role** .

 This instance uses an IAM role named **viewerInstanceProfileForMedicalImages**.

1. Do **not** make a change at this time. Select the **Cancel** button.

Verify the kind of permissions this role is assigning to the viewer instance.

1. At the top of the AWS Management Console, in the search bar, search for and choose

IAM

.

1. In the left navigation pane, select **Roles**.
2. In the Search box, type

viewer

.

1. Select the blue link for **viewerInstanceRoleForMedicalImages**.
2. In the Permissions tab, expand the  **viewerInstancePolicySettingsForMedicalImages** policy.
3. Scroll through the policy text, looking for the **StringNotEqualsIgnoreCase** condition. Observe that the upper portion of this policy allows *encryption* without exception, but the lower section only allows *decryption* using a KMS key that is tagged with an **ImageType** of **XRAY**. It explicitly denies decryption requests where this key is not present.

 Recall that when you created the KMS key, you configured it for access to **MRI** images, not **XRAY** images. Let’s reconfigure it to allow use of XRAY images from our viewer instance.

1. At the top of the AWS Management Console, in the search bar, search for and choose

KMS

.

1. Select **Key Management Service** from the list of Services displayed.
2. Select the blue text link for **ImageAccessKey**.
3. Select the Tags tab.
4. Select the **Edit** button.
5. Replace tag value MRI with

XRAY

 (**Note**: tag names and values are Case Sensitive).

1. Select **Save** .

**Expected service output:**

**Tags updated**

 Without needing to edit an IAM policy or changing any role assignments in EC2, you have just used *attribute-based access control* to grant the ability to decrypt **XRAY images** for the medical technicians working on their EC2 instance.

Verify the change has taken place.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Launch the Image Viewer tool using the **View Images** desktop shortcut. You need the name of the **ImageStorageBucket** that appears in the AWS CONSOLE INFORMATION area to the left of these instructions.
3. Follow the prompts to view one or more of the **Image1.jpg**, **Image2.jpg**, etc. images in the S3 bucket. Verify that this task succeeds and that you are able to view the XRay image.

*Note: all images used in this lab are public domain images available under Creative Commons licensing.*

1. When the Image Viewer tool says: “Display Another (Y/N)”, type

N

 followed by the Enter key, close the Windows PowerShell window that contains this tool, returning to the Windows desktop.

**Congratulations!** You successfully reconfigured the settings to allow a medical technician to access those XRays.

**Task 5: Use the DynamoDB Encryption Client to encrypt database data in DynamoDB**

Another service that benefits from encryption is the DynamoDB database service. A variety of secure storage options are available to the user of this flexible service, including built-in server-side encryption. In this task, you enhance tje security posture with a couple of different varieties of client-side encryption. Implementing client-side encryption adds complexity to a project, but may be a requirement of some workloads. Write some Python code that will implement this capability.

1. Launch the PyCharm app using the desktop shortcut above the Transfer Images shortcut.
2. When the PyCharm User Agreement window appears, select the checkbox to accept the PyCharm Terms of Use and select the **Continue** button.
3. Select the **Don’t Send** button in the Data Sharing window.
4. In the Welcome to PyCharm (Administrator) window, select the Open button.
5. In the Open dialog box, select the **SpotlightLabProjects** folder and select the Ok button.
6. In the Tip of the Day dialog box, select the **Don’t show tips** checkbox and the **Close** button.
7. In the Code with me box in the upper right, select the **Got it** button.
8. In the left Project pane, Select the SalesAnalytics.py file, and type the Shift+Enter key combination to Open the script in the Right Split.

This file contains placeholders for the programming instructions for implementing advanced encryption capabilities. You add the instructions to make it functional in these following steps.

For convenience, this lab includes all the necessary instructions in a copy-and-paste format - you do not need to type any Python programming yourself.

 It is important to take some time to understand what these programming instructions are doing. Be sure to think through each piece of the code you are copying into place, and understand how it contributes to the security of the data that it is storing.

1. Copy and paste the following code into the SalesAnalytics.py script in the area labeled **# ToDo-1 - Identify AWS region and build Table 1**.

 You can easily navigate to ToDo sections using the TODO feature in the bottom-left corner of the PyCharm interface.

# Identify current AWS infrastructure Region

currentAWSRegion = (requests.get('http://169.254.169.254/latest/meta-data/placement/region')).text

# Build connection to DynamoDB service in the current region

dynamodb\_client = boto3.client('dynamodb', currentAWSRegion)

#

if 'Table1' not in dynamodb\_client.list\_tables()['TableNames']:

spotlightLabTable1 = create\_spotlight\_lab\_table('Table1')

These instructions learn which AWS Infrastructure Region the instructions are running in, and build the first demonstration table in DynamoDB. These Python language instructions use some modules that supply add-on functionality needed. The next step imports the appropriate modules.

1. Copy and paste the following code into the area labeled **# ToDo-2 Import Python Modules and Classes**.

 ToDo-2 is at the top of the SalesAnalytics.py script - you will need to scroll up to find ToDo-2.

import boto3

import requests

The code in **ToDo-1** also calls on a custom function to create DynamoDB tables of the type we need for this project. The next step implements that function.

1. Copy and paste the following code into the area labeled **# ToDo-3 Implement table-creation function**.

def create\_spotlight\_lab\_table(new\_table\_name):

print(f'Creating DynamoDB table {new\_table\_name}...')

# creating the table with two schema (main) attributes: OrderID and Shipping Date

table = dynamodb\_client.create\_table(

TableName=new\_table\_name,

KeySchema=[

{

'AttributeName': 'Order ID',

'KeyType': 'HASH' # Partition key

},

{

'AttributeName': 'Ship Date',

'KeyType': 'RANGE' # Sort key

}

], # Both schema elements are datatype 'string' - 'S'

AttributeDefinitions=[

{

'AttributeName': 'Order ID',

'AttributeType': 'S'

},

{

'AttributeName': 'Ship Date',

'AttributeType': 'S'

}

],

BillingMode='PAY\_PER\_REQUEST' # Using the 'fire and forget' Pay Per Request pricing strategy

)

waiter = dynamodb\_client.get\_waiter('table\_exists')

waiter.wait(TableName=new\_table\_name) # Waiting for the table to finish being created

print(f'{new\_table\_name} creation complete.')

return table

This is enough Python code to create a table for the first DynamoDB encryption test. Run this program and verify that **Table1** was created in the DynamoDB web console.

1. At the top of the PyCharm app, Select the Run menu, and then select  **Run…**
2. In the Run dialog box, select **2  SalesAnalytics**.
3. The script begins running. Table creation may take several seconds.

**Expected output:**

C:\Python310\python.exe C:/SpotlightLabProjects/SalesAnalytics.py

Creating DynamoDB table Table1...

Table1 creation complete.

Process finished with exit code 0

1. At the top of the AWS Management Console, in the search bar, search for and choose

DynamoDB

.

1. In the left navigation area, select **Tables**.
2. Verify that **Table1** appears in the console.
3. Select the blue link for **Table1**.
4. In the Overview tab, in the General Information section, expand  **Additional info** and verify that **Encryption** is set to **Owned by Amazon**.

DynamoDB automatically delivers server-side encryption of data, though you get to choose which variety of encryption makes the most sense for your workload.

Next, upload some data to this table and verify that it is visible in the console in plain text.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Copy and paste the following code into the area labeled **# ToDo-4 -Load Table 1 with sales data**. You need to scroll **down** through the script to locate **ToDo-4**.

 You can easily navigate to the ToDo sections using the TODO feature in the bottom-left corner of the PyCharm interface.

#Connect to DynamoDB service

dynamodb\_resource = boto3.resource('dynamodb', currentAWSRegion)

#Open Sales Data file

with open('c:\\temp\\50000 Sales Records.csv', 'rt') as f:

salesDataReader = csv.DictReader(f)

itemCount=0

if dynamodb\_resource.Table('Table1').scan()['ScannedCount'] == 0: #Wasteful action - don't do this in real life!.

for salesRow in salesDataReader:

if salesRow['Item Type'] == 'Office Supplies' and salesRow['Region'] == 'Europe':

itemCount = itemCount + 1

# Insert European Office Supply records into Table 1

dynamodb\_resource.Table('Table1').put\_item(

Item={

'Region': salesRow['Region'],

'Country': salesRow['Country'],

'Item Type': salesRow['Item Type'],

'Sales Channel': salesRow['Sales Channel'],

'Order Priority': salesRow['Order Priority'],

'Order Date': salesRow['Order Date'],

'Order ID': salesRow['Order ID'],

'Ship Date': salesRow['Ship Date'],

'Units Sold': salesRow['Units Sold'],

'Unit Price': salesRow['Unit Price'],

'Unit Cost': salesRow['Unit Cost'],

'Total Revenue': salesRow['Total Revenue'],

'Total Cost': salesRow['Total Cost'],

'Total Profit': salesRow['Total Profit']

}

)

print(salesRow)

print(f"\nInserted {itemCount} sales records into Table1")

These instructions connect to the DynamoDB service, read a CSV (comma-separated values) file looking for sales of office supplies to Europe, and insert those data items into **Table1** in DynamoDB. **Note** the use of the DynamoDB command called **PutItem**. This instruction sends a row of data (called an Item in DynamoDB) to the service for storage.

These Python language instructions use another add-on module. The next step imports that module.

1. Copy and paste the following code into the area at the top of SalesAnalytics.py labeled **# ToDo-2 Import Python Modules and Classes**, making sure not to remove any of the existing import statements. The order of the import statements does not matter. You need to scroll **up** to the top of the script to locate **ToDo-2**.

import csv

Run this program and verify that **Table1** has been populated with sales data.

1. At the top of the PyCharm app, Select the Run menu, and at the top of that menu, select  **Run SalesAnalytics**.

 You no longer use the lower **Run…** option that brings up the secondary dialog box you saw earlier.

1. Verify that the Run pane at the bottom of the PyCharm app display a list of sales records ending with the following:

{'Region': 'Europe', 'Country': 'United Kingdom', 'Item Type': 'Office Supplies', 'Sales Channel': 'Online', 'Order Priority': 'H', 'Order Date': '8/3/2014', 'Order ID': '399203956', 'Ship Date': '9/19/2014', 'Units Sold': '2177', 'Unit Price': '651.21', 'Unit Cost': '524.96', 'Total Revenue': '1417684.17', 'Total Cost': '1142837.92', 'Total Profit': '274846.25'}

Inserted 1033 sales records into Table1

Process finished with exit code 0

1. At the top of the AWS Management Console, in the search bar, search for and choose

DynamoDB

.

1. In the left navigation area, select **Tables**.
2. Select the blue link for **Table1**.
3. In the Overview tab, scroll down to the **Items summary** section. Select the **Get live item count** button.
4. Select the **Start scan** button.
5. Verify that **1033 items** have been stored in the table.
6. Select the **Cancel** button.
7. Choose **Explore table items** .
8. Under the **Items returned** section, verify that all attributes are readable for the displayed items.

The items you are seeing represent encrypted data that you stored for this table in DynamoDB. DynamoDB Server-Side Encryption automatically decrypted those items in response to your request to view them.

This is an easy way to implement an encryption-at-rest requirement for a workload. This default behavior of DynamoDB tables minimizes the risk of misplaced keys, or of data being stored on disk in an unencrypted form, and does so with no administrative effort. Other Server-Side Encryption options provide more control and visibility at a cost of increased effort.

The next series of tasks will explore two different techniques for implementing Client-Side Encryption using the DynamoDB Encryption SDK. In these tasks, you trigger encryption of data items on the local computer first, and only then will the resulting encrypted data be sent to DynamoDB for storage. These configurations implement an additional layer of encryption security, with trade-offs of additional complexity and increased risk of loss of the keys that are necessary to decrypt the data.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Copy and paste the following code into the area labeled **# ToDo-5 - Create Table2**. You need to scroll **down** to locate **ToDo-5**.

#Connect to DynamoDB service and verify Table creation

dynamodb\_resource = boto3.resource('dynamodb', currentAWSRegion)

#Create table 2 with helper table-creation function

if 'Table2' not in dynamodb\_client.list\_tables()['TableNames']:

spotlightLabTable2 = create\_spotlight\_lab\_table('Table2')

else:

spotlightLabTable2 = dynamodb\_resource.Table('Table2')

dynamodb\_resource = boto3.resource('dynamodb', currentAWSRegion)

1. At the top of the PyCharm app, Select the Run menu, and then select  **Run SalesAnalytics** from the top of the menu.

**Expected output:**

Creating DynamoDB table Table2...

Table2 creation complete.

Process finished with exit code 0

1. If you wish, verify the creation of **Table2** in the DynamoDB console, and then continue with the next step.
2. Copy and paste the following code into the area labeled **# ToDo-6 - Load Table2 with client-encrypted sales data using DynamoDB Encryption SDK and KMS Keys**

 You can easily navigate to ToDo sections using the TODO feature in the bottom-left corner of the PyCharm interface.

# Open Sales Data file

with open('c:\\temp\\50000 Sales Records.csv', 'rt') as f:

salesDataReader = csv.DictReader(f)

itemCount=0

# Create KMS Cryptographic Materials Provider

spotlight\_lab\_direct\_kms\_cmp = AwsKmsCryptographicMaterialsProvider(

'replace\_these\_words\_with\_KMS\_ARN')

# Create EncryptedTable object using table and CMP

encrypted\_table\_access = EncryptedTable(

table=spotlightLabTable2,

materials\_provider=spotlight\_lab\_direct\_kms\_cmp

)

if dynamodb\_resource.Table('Table2').scan()['ScannedCount'] == 0: #Wasteful action - don't do this in real life!.

for salesRow in salesDataReader:

if int(salesRow['Units Sold']) < 1000 and salesRow['Region'] == 'North America':

print(salesRow)

itemCount = itemCount + 1

# Issue item storage request to EncryptedTable object

encrypted\_table\_access.put\_item(

Item={

'Region': salesRow['Region'],

'Country': salesRow['Country'],

'Item Type': salesRow['Item Type'],

'Sales Channel': salesRow['Sales Channel'],

'Order Priority': salesRow['Order Priority'],

'Order Date': salesRow['Order Date'],

'Order ID': salesRow['Order ID'],

'Ship Date': salesRow['Ship Date'],

'Units Sold': salesRow['Units Sold'],

'Unit Price': salesRow['Unit Price'],

'Unit Cost': salesRow['Unit Cost'],

'Total Revenue': salesRow['Total Revenue'],

'Total Cost': salesRow['Total Cost'],

'Total Profit': salesRow['Total Profit']

}

)

print(f'\nInserted {itemCount} sales records into Table2')

This code uses more module components. The next step imports what’s needed.

1. Copy and paste the following code into the area at the top of SalesAnalytics.py labeled **# ToDo-2 Import Python Modules and Classes**, making sure not to remove any of the existing import statements. The order of the import statements does not matter.

 ToDo-2 is located at the top of the script.

from dynamodb\_encryption\_sdk.encrypted.table import EncryptedTable

from dynamodb\_encryption\_sdk.material\_providers.aws\_kms import AwsKmsCryptographicMaterialsProvider

The large body of code you recently added in **ToDo-6** includes an instruction to utilize an AWS KMS key to create a resource called a Cryptographic Materials Provider (CMP). A KMS Key needs to be created and configured for this step to succeed. Create that new key.

1. At the top of the AWS Management Console, in the search bar, search for and choose

KMS

.

1. Select **Key Management Service** from the list of Services displayed.
2. Select **Create a key** .
3. Select Key type:

Symmetric

.

1. Under Advanced options, verify the following:

* Key material origin:

KMS

* Regionality:

Single-Region key

1. Select **Next** .
2. Specify the following settings:

* Alias:

DynamoDB\_Direct\_CMP\_Key

* Description:

This key is used to encrypt and decrypt sales data using the DynamoDB Encryption SDK.

 You may need to scroll up to see the Alias field.

1. Select **Next** .
2. In **Step 3: Define key administrative permissions**, under **Key administrators**, in the search box with the  icon, type

user

.

1. Select  the user or role you’re signed into the Console with.

This user is displayed at the top of the page, to the right of the region.

1. Select **Next** .
2. In **Step 4: Define key usage permissions**, under **This account**, search  for

view

 and select the checkbox next to **viewerInstanceRoleForMedicalImages**.

1. Select **Next** .
2. Select **Finish** .

**Expected service output:**

**Success**

1. In the list of **Customer managed keys**, select the blue link named **DynamoDB\_Direct\_CMP\_Key**.
2. In the **General configuration** area, copy the ARN to the clipboard using the  button. You will need this value in the next task.
3. Go back to the RDP with Fleet Manager Connection tab.
4. Paste the copied ARN from the previous step into the script in place of the

replace\_these\_words\_with\_KMS\_ARN

 text contained within the ToDo-6 block. **Important**: do not remove the single quotes that surround the text. You will need to scroll down through the script to locate ToDo-6.

Run this program and verify that Table2 has been populated with client-encrypted data.

 Most of what you have sent to **Table 2** is now **not** be human readable unless decrypted first. The exception is the two primary key fields, which must remain decrypted in order for DynamoDB to locate data in response to queries.

1. At the top of the PyCharm app, Select the Run menu, and at the top of that menu, select  **Run SalesAnalytics**.

 You no longer use the lower **Run…** option that brings up the secondary dialog box you saw earlier.

1. Verify that the Run pane at the bottom of the PyCharm app displays a list of sales records ending with the following:

{'Region': 'North America', 'Country': 'Mexico', 'Item Type': 'Snacks', 'Sales Channel': 'Online', 'Order Priority': 'M', 'Order Date': '1/13/2010', 'Order ID': '225233304', 'Ship Date': '1/13/2010', 'Units Sold': '395', 'Unit Price': '152.58', 'Unit Cost': '97.44', 'Total Revenue': '60269.10', 'Total Cost': '38488.80', 'Total Profit': '21780.30'}

Inserted 93 sales records into Table2

Process finished with exit code 0

1. At the top of the AWS Management Console, in the search bar, search for and choose

DynamoDB

.

1. In the left navigation pane, select **Tables**.
2. Select the blue link for **Table2**.
3. In the Overview tab, scroll down to the **Items summary** section. Select the **Get live item count** button.
4. Select the **Start scan** button.
5. Verify that **93 items** have been stored in the table.
6. Choose **Explore table items** .
7. Verify that only the first two attributes are readable for the displayed items. The remaining attributes remain encrypted. Also note that the DynamoDB Encryption SDK has added two additional attributes, **\*amzn-ddb-map-desc\*** and **\*amzn-ddb-map-sig\***.

 The items you are seeing contain client-side-encrypted data. DynamoDB is unable to decrypt these items. These items can only be decrypted by client-side code like what you used to create these encrypted resources.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Copy and paste the following code into the area labeled **# ToDo-7 - Create Table 3 and use the DynamoDB Encryption Client with a Wrapped Materials Provider**.

#Connect to DynamoDB service and verify Table creation

dynamodb\_resource = boto3.resource('dynamodb', currentAWSRegion)

#Create table 3 with helper table-creation function

if 'Table3' not in dynamodb\_client.list\_tables()['TableNames']:

spotlightLabTable3 = create\_spotlight\_lab\_table('Table3')

else:

spotlightLabTable3 = dynamodb\_resource.Table('Table3')

dynamodb\_resource = boto3.resource('dynamodb', currentAWSRegion)

# Open Sales Data file

with open('c:\\temp\\50000 Sales Records.csv', 'rt') as f:

salesDataReader = csv.DictReader(f)

itemCount=0

# Retrieve encryption keys from corporate key management system on premises

private\_wrapping\_key = get\_wrapping\_key\_from\_on\_prem\_key\_management\_system()

private\_signing\_key = get\_signing\_key\_from\_on\_prem\_key\_management\_system()

# Assemble keys into a C.M.P.

spotlight\_lab\_wrapped\_cmp = WrappedCryptographicMaterialsProvider(

wrapping\_key=private\_wrapping\_key,

unwrapping\_key=private\_wrapping\_key,

signing\_key=private\_signing\_key

)

# Create EncryptedTable object by supplying table and CMP

encrypted\_table\_access = EncryptedTable(

table=dynamodb\_resource.Table('Table3'),

materials\_provider=spotlight\_lab\_wrapped\_cmp

)

# Load data about low-revenue vegetable sales into table 3

if dynamodb\_resource.Table('Table3').scan()['ScannedCount'] == 0: #Wasteful action - don't do this in real life!.

for salesRow in salesDataReader:

if float(salesRow['Total Revenue']) < 500000 and salesRow['Item Type'] == 'Vegetables':

print(salesRow)

itemCount = itemCount + 1

# Issue item storage request to EncryptedTable object

encrypted\_table\_access.put\_item(

Item={

'Region': salesRow['Region'],

'Country': salesRow['Country'],

'Item Type': salesRow['Item Type'],

'Sales Channel': salesRow['Sales Channel'],

'Order Priority': salesRow['Order Priority'],

'Order Date': salesRow['Order Date'],

'Order ID': salesRow['Order ID'],

'Ship Date': salesRow['Ship Date'],

'Units Sold': salesRow['Units Sold'],

'Unit Price': salesRow['Unit Price'],

'Unit Cost': salesRow['Unit Cost'],

'Total Revenue': salesRow['Total Revenue'],

'Total Cost': salesRow['Total Cost'],

'Total Profit': salesRow['Total Profit']

}

)

print(f'\nInserted {itemCount} sales records into Table3')

This code uses more module components. The next step imports what’s needed.

1. Copy and paste the following code into the area at the top of SalesAnalytics.py labeled **# ToDo-2 Import Python Modules and Classes**, making sure not to remove any of the existing import statements. The order of the import statements does not matter.

 You need to scroll to the top of the script to locate **ToDo-2**.

from dynamodb\_encryption\_sdk.material\_providers.wrapped import WrappedCryptographicMaterialsProvider

from dynamodb\_encryption\_sdk.delegated\_keys.jce import JceNameLocalDelegatedKey

from dynamodb\_encryption\_sdk.identifiers import EncryptionKeyType, KeyEncodingType

from Crypto.Random import get\_random\_bytes

The code in this section also calls on two custom functions to retrieve encryption keys from an on premises key management system.

 This is *simulated* behavior - no actual connection is being made to an external system.

1. Copy and paste the following code into the area labeled **# ToDo-8 Implement on-prem key retrieval functions**.

**ToDo-8** appears shortly after **ToDo-3**.

def get\_wrapping\_key\_from\_on\_prem\_key\_management\_system():

# Don't implement your program this way!

# I'm pretending that we're getting keys from an on-prem system

# for the purposes of this lab. But I'm actually generating them locally

new\_wrapping\_key = JceNameLocalDelegatedKey(

key=get\_random\_bytes(32),

algorithm='AES',

key\_type=EncryptionKeyType.SYMMETRIC,

key\_encoding=KeyEncodingType.RAW,

)

return new\_wrapping\_key

def get\_signing\_key\_from\_on\_prem\_key\_management\_system():

# Don't implement your program this way!

# I'm pretending that we're getting keys from an on-prem system

# for the purposes of this lab. But I'm actually generating them locally

new\_signing\_key = JceNameLocalDelegatedKey(

key=get\_random\_bytes(32),

algorithm='HmacSHA512',

key\_type=EncryptionKeyType.SYMMETRIC,

key\_encoding=KeyEncodingType.RAW,

)

return new\_signing\_key

Run this program and verify that **Table3** gets populated with client-encrypted data.

 Most of what has been sent to **Table3** is now **not** be human readable unless decrypted first. The exception is the two primary key fields, which must remain decrypted in order for DynamoDB to locate data in response to queries.

1. At the top of the PyCharm app, Select the Run menu, and at the top of that menu, select  **Run SalesAnalytics**.

 You no longer use the lower **Run…** option that brings up the secondary dialog box you saw earlier.

1. Verify that the Run pane at the bottom of the PyCharm app display a list of sales records ending with the following:

{'Region': 'Central America and the Caribbean', 'Country': 'Honduras', 'Item Type': 'Vegetables', 'Sales Channel': 'Online', 'Order Priority': 'H', 'Order Date': '6/7/2012', 'Order ID': '563878285', 'Ship Date': '7/22/2012', 'Units Sold': '2970', 'Unit Price': '154.06', 'Unit Cost': '90.93', 'Total Revenue': '457558.20', 'Total Cost': '270062.10', 'Total Profit': '187496.10'}

Inserted 1345 sales records into Table3

Process finished with exit code 0

1. At the top of the AWS Management Console, in the search bar, search for and choose

DynamoDB

.

1. In the left navigation area, select **Tables**.
2. If you do not see the list for **Table3**, choose the refresh  button.
3. Select the blue link for **Table3**.
4. In the Overview tab, scroll down to the **Items summary** section. Select the **Get live item count** button.
5. Select the **Start scan** button.
6. Verify that 1345 items have been stored in the table.
7. Select the **Cancel** button.
8. Choose **Explore table items** .
9. Verify that, like the previous scenario, only the first two attributes are readable for the displayed items. The remaining attributes remain encrypted until actively decrypted through use of keys retrieved from your on-prem system. Also note that the DynamoDB Encryption SDK has added two additional attributes, **\*amzn-ddb-map-desc\*** and **\*amzn-ddb-map-sig\*** to help manage the encryption.

The items you are seeing contain client-side-encrypted data. DynamoDB is unable to decrypt these items. These items can only be decrypted by client-side code like what you used to create these encrypted resources.

**Congratulations!** You used the DynamoDB Encryption Client to encrypt database data in DynamoDB for multiple tables.

**Task 6: (Optional - if time permits) Use the AWS Encryption SDK**

In this task, it is time to tackle the most important encryption challenge - protecting your grandmother’s famous family recipes. So many aunts and uncles are itching to get their hands on them. If only AWS provided a way to keep their prying eyes out.

Take a look at the contents of the recipe files as they stand currently.

1. Go back to the RDP with Fleet Manager Connection tab.
2. Open the Windows File Explorer using the yellow taskbar icon within your RDP with Fleet Manager Connection tab.
3. Open the **C:\Personal\Secret Family Recipes** folder.
4. Open each of grandma’s delicious recipes and confirm that they are all in plaintext.
5. Switch to the PyCharm app.
6. Copy and paste the following code into the area labeled **# ToDo-9 Use AWS Encryption SDK to protect local data**.

# Collect files from Grandma's secret recipes folder

recipes\_directory = 'C:\\Personal\\Secret Family Recipes\\'

recipe\_file\_list = listdir(recipes\_directory)

for recipe in recipe\_file\_list:

print('Encrypting ' + recipe)

# Open a recipe file and read the text inside

recipe\_file = open(recipes\_directory + recipe)

recipe\_text = recipe\_file.read()

recipe\_file.close()

# Create the AWS Encryption SDK client and attach it to a KMS Key

client = aws\_encryption\_sdk.EncryptionSDKClient(commitment\_policy=CommitmentPolicy.REQUIRE\_ENCRYPT\_REQUIRE\_DECRYPT)

keys = dict(key\_ids=['replace\_these\_words\_with\_KMS\_ARN'])

master\_key\_provider = aws\_encryption\_sdk.StrictAwsKmsMasterKeyProvider(\*\*keys)

# Transform plain recipe text into encrypted recipe text

ciphertext, encryptor\_header = client.encrypt(source=recipe\_text, key\_provider=master\_key\_provider)

# Store encrypted recipe bytes into new file

new\_recipe\_file = open(recipes\_directory + '[ENCRYPTED]-' + recipe, "wb")

encrypted\_file\_content = bytearray(ciphertext)

new\_recipe\_file.write(encrypted\_file\_content)

new\_recipe\_file.close()

print(' ... done.')

This code uses one last batch of new module components. Here is the final set of import statements.

1. Copy and paste the following code into the area at the top of SalesAnalytics.py labeled **# ToDo-2 Import Python Modules and Classes**, making sure not to remove any of the existing import statements. The order of the import statements does not matter.

import aws\_encryption\_sdk

from aws\_encryption\_sdk import CommitmentPolicy

from os import listdir

The AWS Encryption SDK allows you to encrypt any arbitrary data of your choosing with the help of an AWS KMS key. A KMS Key needs to be created and configured for this step to succeed. Create that new key next.

1. At the top of the AWS Management Console, in the search bar, search for and choose

KMS

.

1. Select **Create a key** .
2. Select Key type:

Symmetric

.

1. Under Advanced options, verify the following:

* Key material origin:

KMS

* Regionality:

Single-Region key

1. Select **Next** .
2. Specify the following settings:

* Alias:

AWS\_Encryption\_SDK\_Key

* Description:

This key is used to encrypt and decrypt grandma's recipes using the AWS Encryption SDK.

  You may need to scroll up to see the Alias field.

1. Select **Next** .
2. In **Step 3: Define key administrative permissions**, under **Key administrators**, in the search box with the  icon, type

user

.

1. Select  the user or role you’re signed into the Console with.

This user is displayed at the top of the page, to the right of the region.

1. Select **Next** .
2. In **Step 4: Define key usage permissions**, under **This account**, search  for

view

 and select the checkbox next to **viewerInstanceRoleForMedicalImages**.

1. Select **Next** .
2. Select **Finish** .
3. In the list of **Customer managed keys**, select the blue link named **AWS\_Encryption\_SDK\_Key**.
4. In the **General configuration** area, copy the ARN to the clipboard using the  button. You will need this value in the next task.
5. Go back to the RDP with Fleet Manager Connection tab.
6. Paste the copied ARN from the previous step into the script in place of the **replace\_these\_words\_with\_KMS\_ARN** text in ToDo-9, near the bottom of the script.

**Caution:** Do not remove the single quotes that surround the text.

Run this program and verify that Grandma’s beloved recipes are kept safe from Uncle Frank’s depredations.

1. At the top of the PyCharm app, Select the Run menu, and at the top of that menu, select  **Run SalesAnalytics**.

 You no longer use the lower **Run…** option that brings up the secondary dialog box you saw earlier.

1. Verify that the Run pane at the bottom of the PyCharm app display a list of sales records ending with the following:

C:\Python310\python.exe C:/SpotlightLabProjects/SalesAnalytics.py

Encrypting Chicken Terrapin with Calves' Brains.txt

... done.

Encrypting Ox-cheek Soup.txt

... done.

Encrypting Roast Lamb with Banana Croquettes.txt

... done.

Process finished with exit code 0

1. Open the Windows File Explorer within the RDP with Fleet Manager Connection tab.
2. Open the C:\Personal\Secret Family Recipes folder
3. Open the new files (marked with [ENCRYPTED]) and verify they are all in in an encrypted binary format.

The KMS Key used to protect these files is well protected by the Key Management Services. But consider that if a user were to delete the key, these delightful recipes could potentially be lost forever!

**Congratulations!** You have successfully encrypted your grandmother’s family recipes.

**Conclusion**

**Congratulations!** You now have successfully:

* Created a KMS Key for encryption management
* Enabled S3 Server-side Encryption
* Configured Attribute-based Access Control
* Used the DynamoDB Encryption Client to encrypt database data in DynamoDB
* Encrypted data programmatically with the AWS Encryption SDK

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

* For more information about how to use AWS KMS, see [AWS KMS Documentation](https://docs.aws.amazon.com/kms/index.html).
* For more information about how to use Amazon S3, see [AWS S3 Documentation](https://docs.aws.amazon.com/s3).
* For more information about how to use Amazon DynamoDB Encryption Client, see [Amazon DynamoDB Encryption Client Documentation](https://docs.aws.amazon.com/dynamodb-encryption-client/latest/devguide/what-is-ddb-encrypt.html).
* For more information about how to use AWS Encryption SDK, see [AWS Encryption SDK Documentation](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html).

**Connect to your Windows instance using RDP**

* To the left of the instructions you are currently reading, choose **Download PEM**.
* Save the file to the directory of your choice.
* Open the [Amazon EC2 console](https://console.aws.amazon.com/ec2/)
* In the navigation pane, select **Instances**. Select the **Image Viewer EC2 Instance** and then choose **Connect**.
* On the **Connect to instance** page, choose the **RDP client** tab, and then choose **Get password**.
* Choose **Browse** and navigate to the private key (**.pem**) file you created earlier. Select the file and choose **Open** to copy the entire contents of the file to this window.
* Choose **Decrypt Password**. The console displays the default administrator password for the instance under **Password**, replacing the **Get password** link shown previously. Save the password in a safe place. This password is required to connect to the instance.

FOR YOUR RDP CLIENT, USE THE FOLLOWING DETAILS TO CONNECT

* **Windows instance IP:** Copy and paste the Public IP from the EC2 console.
* **Username:** Enter

**Administrator**

.

* **Password:** Copy and paste the **password** that you saved previously.

[Return to the instructions](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-TF-300-SIDTEN-1%3A1.0.11-684b658d/en-US#bastion)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).