

Cyber Shujaa

Cloud Security Specialist

**Assignment 20 - Lab 7: Key Vault
(Implementing Secure Data by setting up
Always Encrypted)**

NAME: WELDON KIPKIRUI KENEI

CS NO: ADC-CSS02-25027

Introduction

In this lab scenario, we will create a proof-of-concept application that uses the Azure SQL Database's support for Always Encrypted functionality. All of the secrets and keys used in this scenario will be stored in Key Vault. The application should be registered in Microsoft Entra ID to enhance its security posture.

Deploy the base infrastructure from an ARM template

In this step, we will deploy an Azure SQL database and an Azure VM using an existing ARM template, which will automatically install Visual Studio 2019 and SQL Server Management Studio 19 as part of the deployment.

The screenshot shows a split-screen interface. On the left, the Microsoft Azure portal displays the 'Microsoft.Template-20250610023740 | Overview' page, indicating a deployment is in progress. A red box highlights the deployment status message. On the right, a 'Key Vault (Implementing Secure Data by setting up Always Encrypted)' lab client window is open, showing exercise instructions for configuring a Key Vault resource. It includes a note about the East (US) region, a list of tasks, and a task 1 description. The bottom of the client window shows a progress bar at 15% complete.

Configure the Key Vault resource with a key and a secret

In this step, we will configure an Azure Key Vault and add a key and a secret to the Key Vault. Use these commands in the Cloud Shell.

```
$kvName = 'az500kv' + $(Get-Random)  
$location = (Get-AzResourceGroup -ResourceGroupName 'AZ500LAB10-lod52070586').Location  
New-AzKeyVault -VaultName $kvName -ResourceGroupName 'AZ500LAB10-lod52070586' -Location $location -DisableRbacAuthorization
```

The screenshot shows the Azure Cloud Shell interface. On the left, a terminal window displays PowerShell commands for creating a new Key Vault:

```
PS /home/labuser-52070586> New-AzKeyVault -VaultName $kvName -ResourceGroupName 'AZ500LAB10-lod52070586' -Location $location -DisableRbacAuthorization
```

The output of the command shows the configuration of the new Key Vault:

Vault Name	:	az500kv1288603018
Resource Group Name	:	AZ500LAB10-lod52070586
Location	:	eastus
Resource ID	:	/subscriptions/436b8761-e2fc-4df7-9677-cf5e4e6a4272/resourceGroups/AZ500LAB10-lod52070586/providers/Microsoft.KeyVault/vaults/az500kv1288603018
Vault URI	:	https://az500kv1288603018.vault.azure.net/
Tenant ID	:	8eb87a6e-8955-4135-b69d-f19c799ec045
SKU	:	Standard
Enabled For Deployment?	:	False
Enabled For Template Deployment?	:	
Enabled For Disk Encryption?	:	
Enabled For RBAC Authorization?	:	False
Soft Delete Enabled?	:	True
Soft Delete Retention Period (days)	:	90
Purge Protection Enabled?	:	
Public Network Access	:	Enabled

On the right, the Cloud Shell pane shows the task progress and instructions:

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Key Vault (Implementing Secure Data by setting up ...)
1 Hr 3 Min Remaining

Instructions Resources Help 100%

menu in the upper-left corner of the Cloud Shell pane.

3. In the PowerShell session within the Cloud Shell pane, run the following to create an Azure Key Vault in the resource group AZ500LAB10-lod52070586. (If you chose another name for the lab's Resource Group out of Task 1, use that name for this task as well). The Key Vault name must be unique. Remember the name you have chosen. You will need it throughout this lab.

```
powershell  
'az500kv' + $(Get-Random)  
= (Get-AzResourceGroup -ResourceGroup  
Vault -VaultName $kvName -ResourceGro
```

The output of the last command will display the vault name and the vault URI. The vault URI is in the format

17% Tasks Complete

< Previous End >

We will then create the access policies for our key vault from the Azure portal as shown below.

The screenshot shows the Microsoft Azure Key Vault Access Policies page for 'az500kv1288603018'. A red box highlights the 'az500kv1288603018 | Access policies' title bar. Another red box highlights the user 'LabUser-52070586' listed under the 'USER' section. On the right, a sidebar provides instructions for creating access policies, mentioning 'selected' on the Principal blade, selecting a user account, and clicking 'Next'. It also notes that the previous Review + create operation returns to the Access policies page. A progress bar at the bottom indicates 24% tasks complete.

Next, we will run the following to add a software-protected key to the Key Vault:

```
$kv = Get-AzKeyVault -ResourceGroupName 'AZ500LAB10-1od52070586'  
$key = Add-AZKeyVaultKey -VaultName $kv.VaultName -Name 'MyLabKey'  
-Destination 'Software'
```

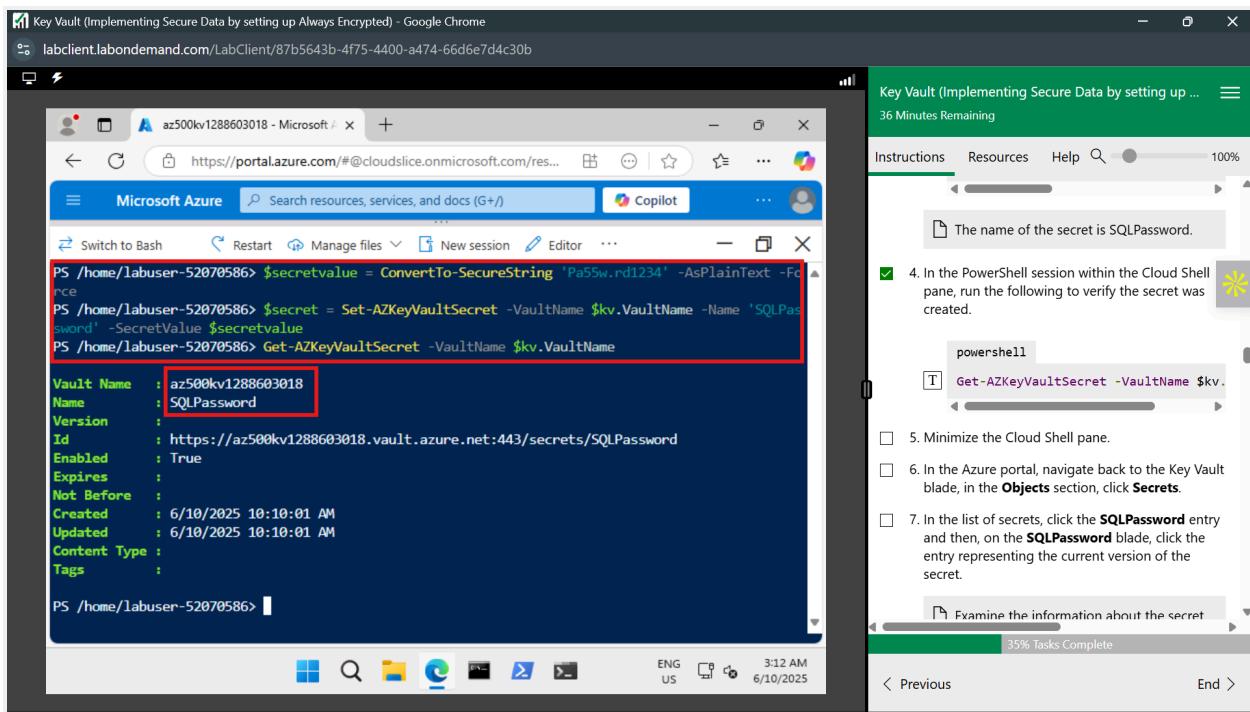
The screenshot shows a Microsoft Azure Cloud Shell session. A red box highlights the PowerShell command 'Get-AZKeyVaultKey -VaultName \$kv.VaultName' being run. Below the session, the properties of the newly created key 'MyLabKey' are displayed in a table. The table includes columns for Vault/HSM Name, Name, Version, Id, Enabled, Expires, Not Before, Created, Updated, Recovery Level, and Tags. The 'Name' column shows 'MyLabKey'. On the right, a sidebar provides instructions for verifying the key creation, running commands in the Cloud Shell, and navigating back to the Azure portal. A progress bar at the bottom indicates 28% tasks complete.

We then add a secret to our KeyVault. Using the Cloud Shell, we run the following commands;

1.

```
$secretvalue = ConvertTo-SecureString 'Pa55w.rd1234'  
-AsPlainText -Force
```
2.

```
$secretvalue = ConvertTo-SecureString 'Pa55w.rd1234' -AsPlainText  
-Force
```



Configure an Azure SQL database and a data-driven application

In this task, we will enable a client application to access the Azure SQL Database service, create a policy allowing the application access to the Key Vault, and retrieve the SQL Azure database ADO.NET Connection String, log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19 ND Create a table in the SQL Database and select data columns for encryption.

First, we will register an application, in this case, sqlapp, as shown below, using the app registration service in the Azure portal.

The screenshot shows a split-screen view of the Azure portal. On the left, a Microsoft Edge browser window displays the 'Register an application' blade. It shows the 'Accounts in any organizational directory' and 'Personal Microsoft accounts only' options selected. A red box highlights the 'Redirect URI (optional)' field, which contains 'Web' and 'https://sqlapp'. Below this, a 'Register' button is highlighted with a red box. On the right, a 'Key Vault (Implementing Secure Data by setting up ...)' blade is open, showing a table with two rows: 'Name' (sqlApp) and 'Redirect URI (optional)' (Web and https://sqlapp). A checkmark is next to step 4, which reads: 'On the Register an application blade, click Register.' Below this, steps 5 and 6 are listed with checkboxes. Step 5 says: 'Once the registration is completed, the browser will automatically redirect you to sqlApp blade.' Step 6 says: 'On the sqlApp blade, identify the value of Application (client) ID.' A note below it says: 'Record this value. You will need it in the next task.' At the bottom of the right blade, it says '42% Tasks Complete'.

We then configure the app secret key as shown below.

The screenshot shows two windows side-by-side. On the left is the Microsoft Azure portal with a sub-menu open for 'Client secrets'. A modal window titled 'Add a client secret' is displayed, with its title bar highlighted by a red box. Inside the modal, there are fields for 'Description' (containing 'Key1') and 'Expires' (set to '365 days (12 months)'). Below these fields is a blue 'Add' button, which is also highlighted by a red box. On the right, a separate window titled 'Key Vault (Implementing Secure Data by setting up Always Encrypted)' displays a task list. Task 8 instructs to specify settings for the client secret, listing 'Description' as 'Key1' and 'Expires' as '12 months'. Task 9 says to click 'Add' to update the application credentials. Task 10 says to identify the value of 'Key1' on the 'Certificates & secrets' blade and to record it. The task list has a progress bar at the bottom indicating '44% tasks Complete'.

Next, we will grant the newly registered app permissions to access secrets stored in the Key Vault. We will run the following to create a variable storing the Application (client) ID we recorded in the previous tasks.

```
$applicationId = '<Azure_AD_Application_ID>'
```

We will run the following to grant permissions on the Key Vault to the application we registered in the previous task:

```
Set-AZKeyVaultAccessPolicy -VaultName $kvName -ResourceGroupName $ResourceGroupName  
AZ500LAB10-lod52070586 -ServicePrincipalName $applicationId  
-PermissionsToKeys get,wrapKey,unwrapKey,sign,verify,list
```

The screenshot shows the Microsoft Azure Cloud Shell interface. On the left, a terminal window displays PowerShell commands:

```
PS /home/labuser-52070586> Set-AZKeyVaultAccessPolicy -VaultName $kvName -ResourceGroupName AZ500LAB10-lod52070586 -ServicePrincipalName $applicationId -PermissionsToKeys get,wrapKey,unwrapKey,sign,verify,list  
PS /home/labuser-52070586>
```

A red box highlights the command being run. On the right, the Key Vault pane shows the configuration for the vault. A task list indicates step 5 is completed:

- 5. In the PowerShell session within the Cloud Shell pane, run the following to grant permissions on the Key Vault to the application you registered in the previous task:

```
powershell  
Set-AZKeyVaultAccessPolicy -VaultName
```

Step 6 is listed as pending:

- 6. Close the Cloud Shell pane.

Task 3: Retrieve SQL Azure database ADO.NET Connection String

The ARM-template deployment in Exercise 1 provisioned an Azure SQL Server instance and an Azure SQL database named **medical**. You will update the empty database resource with a new table structure and select data columns for encryption

53% Tasks Complete

Since we have an empty database provisioned by the ARM template we deployed earlier, we can update it with a new table structure and select data columns for encryption. First, we will retrieve the [ADO.NET](#) string for SQL authentication as shown;

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Microsoft Azure Search resources, services, and docs (G+)

ADO.NET (MICROSOFT Entity passwordless authentication)

Microsoft.Data.SqlClient Quickstart Entity Framework Core Quickstart

```
Server=tcp:sqlservermjv2u4kwrmdsg.database.windows.net,1433;Initial Catalog=medical;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;Authentication="Active Directory Default";
```

ADO.NET (SQL authentication)

```
Server=tcp:sqlservermjv2u4kwrmdsg.database.windows.net,1433;Initial Catalog=medical;Persist Security Info=False;User ID=Student;Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;
```

Download ADO.NET driver for SQL server

Key Vault (Implementing Secure Data by setting up ... 44 Minutes Remaining)
Instructions Resources Help 100%
for ADO.NET, JDBC, ODBC, PHP, and Go.

4. Record the **ADO.NET (SQL authentication)** connection string. You will need it later.
When you use the connection string, make sure to replace the **{your_password}** placeholder with the password that you configured with the deployment in Exercise 1.

Task 4: Log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19
In this task, you log on to the Azure VM, which deployment you initiated in Exercise 1. This Azure VM hosts Visual Studio 2019 and SQL Server Management Studio 19.
Before you proceed with this task, ensure that the deployment you initiated in the first exercise has completed successfully. You can validate this by navigating to the blade of the Azure resources.

57% Tasks Complete

< Previous End >

Now we can configure our SQL database firewall rules to allow our virtual machine that was deployed earlier to publicly access it using the following firewall rules.

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Microsoft Azure Search resources, services, and docs (G+)

Key Vault (Implementing Secure Data by setting up ... 36 Minutes Remaining)
Instructions Resources Help 100%

Firewall rules
Allow certain public internet IP addresses to access your resource. [Learn more](#)

+ Add your client IPv4 address (185.254.59.122) + Add a firewall rule

Add a firewall rule
Rule name: Allow Mgmt VM, Start IP: 52.170.195.73, End IP: 52.170.195.73
OK Cancel Save Discard

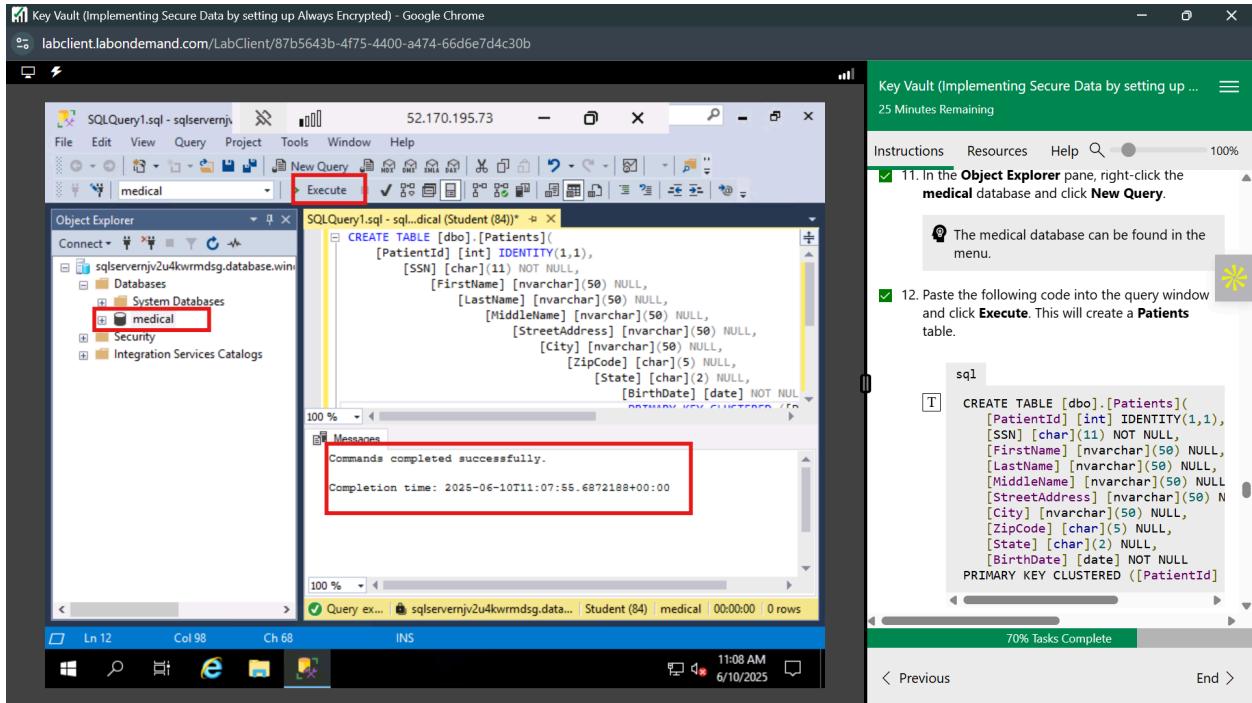
4. Navigate back to the **az500-10-vm1** blade, click **Overview**, next click **Connect** and, in the drop down menu, click **RDP**.
5. Click **Download RDP File** and use it to connect to the **az500-10-vm1** Azure VM via Remote Desktop. When prompted to authenticate, provide the following credentials:

Setting	Value
Username	T Student
Password	T Please use your personal password created in Lab 02 > Exercise 1 > Task 1 > Step 9.

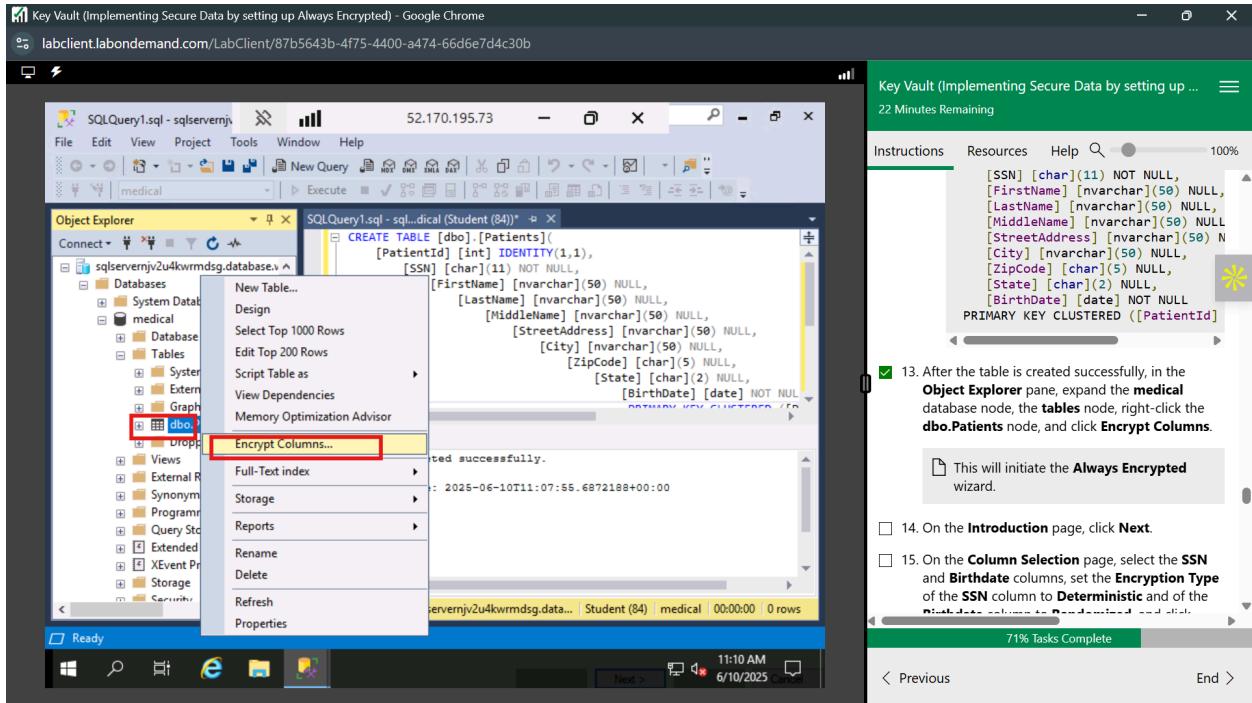
60% Tasks Complete

< Previous End >

From our VM, we can connect to the database using the SQL Server Management Studio app installed. We then interact with our database with queries to update the table structures as shown;



We will select a table and columns in it to always encrypt.



Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Column Selection

Introduction
Enable Secure Enclaves
Column Selection
Master Key Configuration
In-Place Encryption Settings
Run Settings
Summary
Results

Search column name...
Apply one key to all checked columns: CEK_Auto1 (New)

Name	State	Encryption Type	Encryption Key
dbo.Patien...	Patien...		
SSN	Deterministic	CEK_Auto1 (New)	
FirstN...			
LastN...			
Middl...			
Street...			
City			
ZipC...			
State			
Birth...	Randomized	CEK_Auto1 (New)	

Show affected columns only

11:14 AM 6/10/2025

Key Vault (Implementing Secure Data by setting up ...)
18 Minutes Remaining
Instructions Resources Help 100%

15. On the **Column Selection** page, select the **SSN** and **Birthdate** columns, set the **Encryption Type** of the **SSN** column to **Deterministic** and of the **Birthdate** column to **Randomized**, and click **Next**.

While performing the encryption if any error thrown like **Exception has been thrown by the target of an invocation** related to **Rotary**(Microsoft.SqlServer.Management.Se) then make sure the **Key Permission's** values of **Rotation Policy Operations** are **unchecked**, if not in the Azure portal navigate to the **Key Vault > Access Policies > Key Permissions** > Uncheck all the values under the **Rotation Policy Operations** > Under **Privileged Key Operations** > Uncheck **Release**.

16. On the **Master Key Configuration** page, select **Azure Key Vault**, click **Sign in**, when prompted, authenticate by using the same user account you used to provision the Azure Key Vault instance earlier in this lab.

72% Tasks Complete

Next, we configure the master encryption key that is stored outside our database and in the key vault as shown below;

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Master Key Configuration

Introduction
Enable Secure Enclaves
Column Selection
Master Key Configuration
In-Place Encryption Settings
Run Settings
Summary
Results

To generate a new column encryption key, a column master key must be selected to protect it. The column master key is stored outside of the database.

Select column master key:
Auto generate column master key

Select the key store provider:
Windows certificate store

Azure Key Vault

You are signed in as LabUser-52070586@cloudslice.onmicrosoft.com. [Change user](#)

Sign Out

Select a subscription to use:
AZ-500T00-A CSR 2 (436b8761-e2fc-4df7-9677-cf5e4e6a4272)

Select an Azure Key Vault:
az500kv1288603018

13 Minutes Remaining
Instructions Resources Help 100%

appears in the **Select an Azure Key Vault** drop down list, and click **Next**.

17. On the **Run Settings** page, click **Next**.

18. On the **Summary** page, click **Finish** to proceed with the encryption. When prompted, sign in again by using the same user account you used to provision the Azure Key Vault instance earlier in this lab.

19. Once the encryption process is complete, on the **Results** page, click **Close**.

20. In the **SQL Server Management Studio** console, in the **Object Explorer** pane, under the **medical** node, expand the **Security** and **Always Encrypted Keys** subnodes.

The **Always Encrypted Keys** subnode contains the **Column Master Keys** and **Column Encryption Keys** subfolders.

Exercise 4: Demonstrate the use of Azure

75% Tasks Complete

From the result we can see that every section and encryption attempt passed.

The screenshot shows the 'Results' page of the Always Encrypted Wizard. A red box highlights the 'Summary' table, which contains three rows:

Task	Details
Generate new column master key CMK_Auto1 in Azure Key Vault az500kv1...	Passed
Generate new column encryption key CEK_Auto1	Passed
Performing encryption operations	Passed

To the right of the wizard window, a task list is displayed:

- In the Visual Studio console, click the Tools menu, in the drop down menu, click NuGet Package Manager, and, in the cascading menu, click Package Manager Console.
- In the Package Manager Console pane, run the following to install the first required NuGet package:


```
powershell
Install-Package Microsoft.SqlServer.
```
- In the Package Manager Console pane, run the following to install the second required NuGet package:


```
powershell
Install-Package Microsoft.IdentityMo
```

Progress bar: 83% Tasks Complete

The screenshot shows the SQL Server Management Studio (SSMS) Object Explorer and a query results window.

Object Explorer: The 'Always Encrypted Keys' node under the 'Security' category is expanded, showing 'Column Master K' and 'Column Encrypt' sub-nodes.

Query Results: A query has been run to create a table:

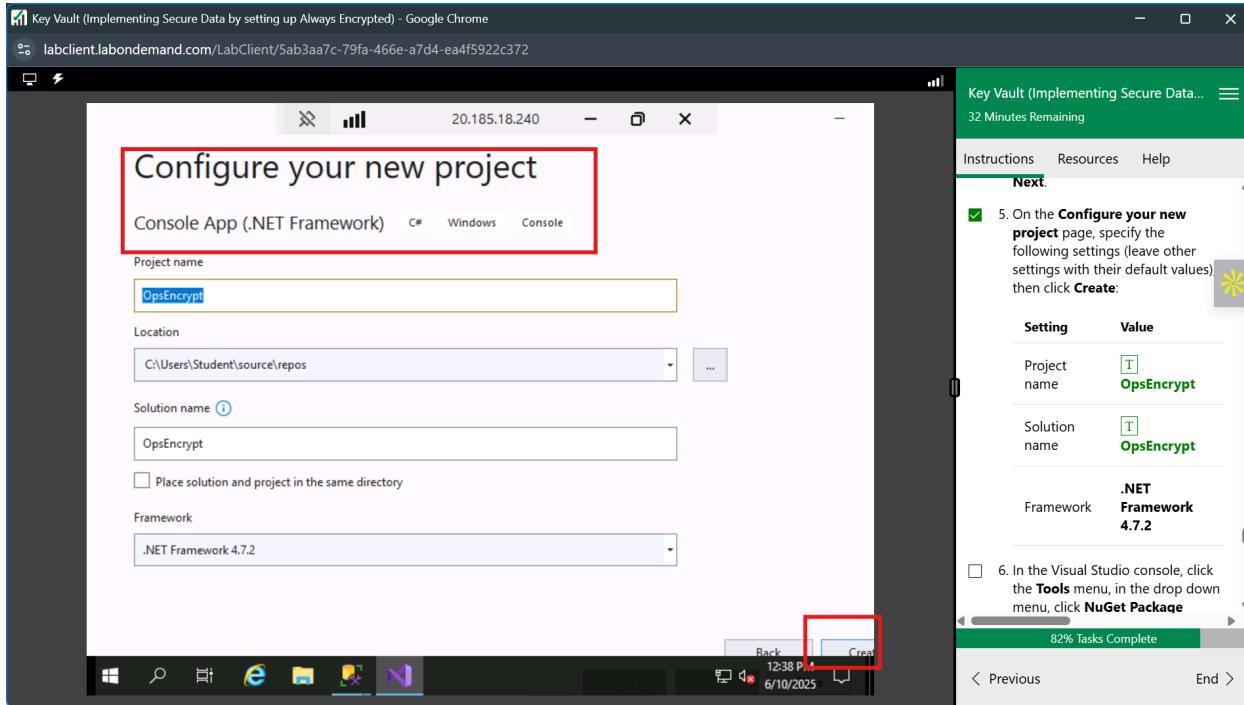
```
CREATE TABLE [dbo].[Patients]
[PatientId] [int] IDENTITY(1,1),
[SSN] [char](11) NOT NULL,
[FirstName] [nvarchar](50) NULL,
[LastName] [nvarchar](50) NULL,
[MiddleName] [nvarchar](50) NULL,
[StreetAddress] [nvarchar](50) NULL,
[City] [nvarchar](50) NULL,
[ZipCode] [char](5) NULL,
[State] [char](2) NULL,
[BirthDate] [date] NOT NULL
```

Message pane: Commands completed successfully.

Exercise 4: Demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

Demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

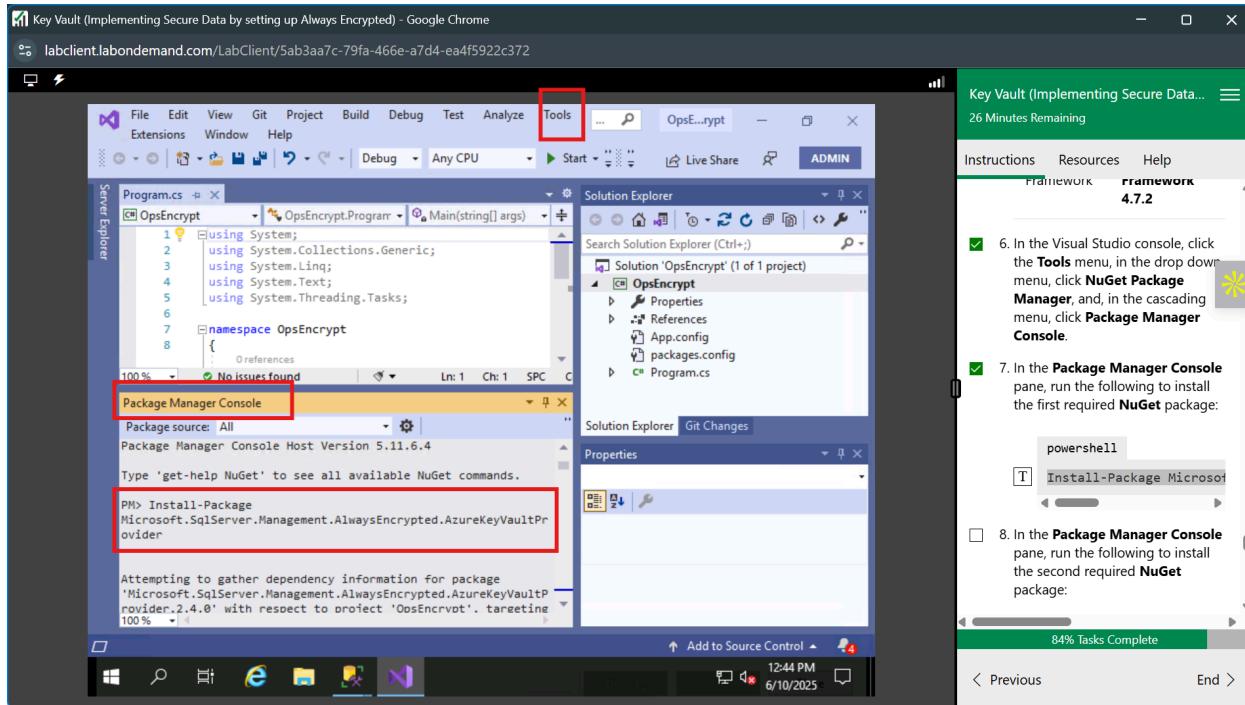
In this task, we will run a data-driven application to demonstrate the use of Azure Key Vault in encrypting the Azure SQL database. We will create a console application using Visual Studio to load data into the encrypted columns and then access that data securely using a connection string that accesses the key in the Key Vault.



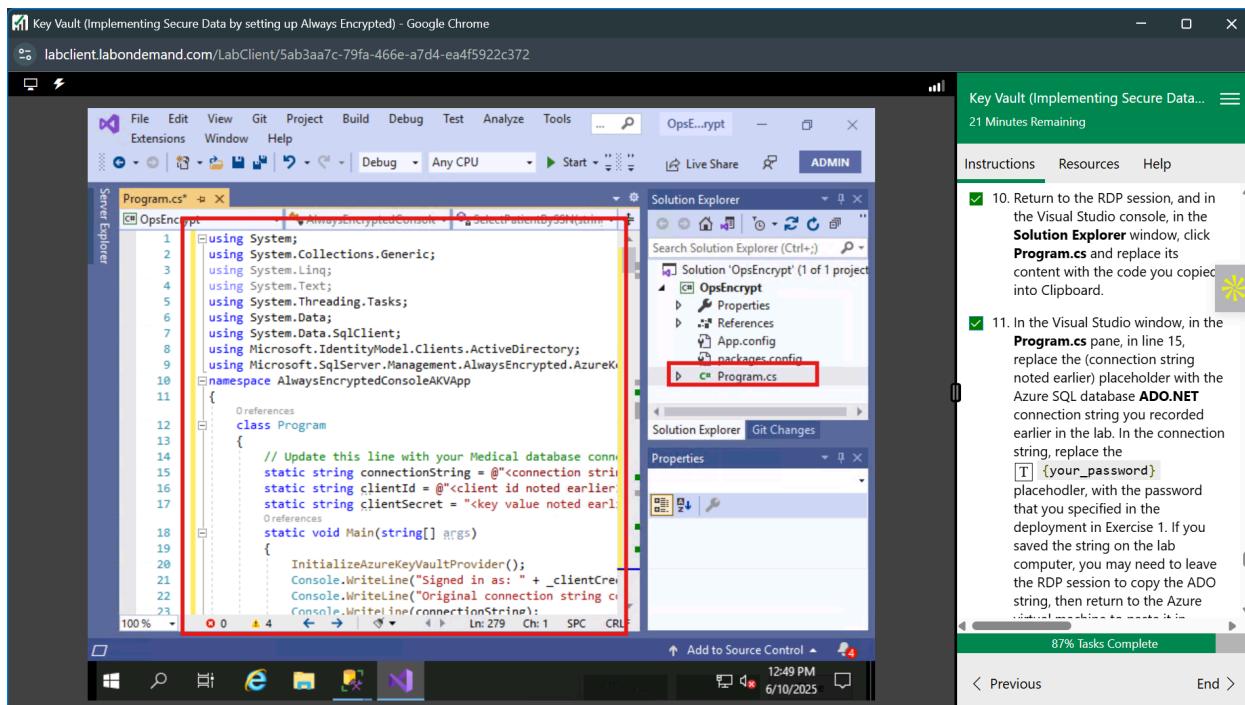
We then navigate to the tools menu and install necessary packages using the following command;

Install-Package

```
Microsoft.SqlServer.Management.AlwaysEncrypted.AzureKeyVaultProvider
```



We then paste the C program code in our lab files into the new Visual Studio window.



From the code, we can replace the ADO.NET connection string in line 15, the client ID of our SQLApp in line 16, and the secret key generated in Key1 in line 17 as shown below;

The screenshot shows a dual-pane interface. On the left is a Microsoft Edge browser window titled "Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome". The URL is "labclient.labondemand.com/LabClient/5ab3aa7c-79fa-466e-a7d4-ea4f5922c372". The right pane is a Microsoft Visual Studio window showing the "Program.cs" file of a project named "AlwaysEncryptedConsoleAKVApp". The code editor highlights several lines of C# code:

```
1 // Collections.Generic;
2 // Linq;
3 // Text;
4 // Threading.Tasks;
5 // Data;
6 // Data.SqlClient;
7 // ft.IdentityModel.Clients.ActiveDirectory;
8 // ft.SqlServer.Management.AlwaysEncrypted.AzureKeyVaultProvider;
9 // EaysEncryptedConsoleAKVApp
10
11
12
13
14 // date this line with your Medical database connection string from the Azure portal.
15 // string connectionString = @"Server=tcp:sqlserverhcjkjkipgiiulfm.database.windows.net,1433;
16 // c string clientId = "sqlserverhcjkjkipgiiulfm.database.windows.net";
17 // c string clientSecret = "IMoBQ-jHjYRu_bgVeB1KivKh7IhmYItfzNzUbDG";
18
19
20
21
22
23 }
```

A red box highlights the connection string line (line 15), the clientId line (line 16), and the clientSecret line (line 17). On the right, the "Task List" pane shows two completed tasks:

- 12. In the Visual Studio window, in the **Program.cs** pane, in line 16, replace the (client id noted earlier) placeholder with the value of **Application (client)** ID of the registered app you recorded earlier in the lab.
- 13. In the Visual Studio window, in the **Program.cs** pane, in line 17,

The status bar at the bottom of the Visual Studio window indicates "88% Tasks Complete".

We then run the code as shown below

The screenshot shows a Microsoft Edge browser window titled "Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome" with the same URL as the previous screenshot. The right pane is a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The command prompt shows the output of the C# application:

```
Signed in as: sqlserverhcjkjkipgiiulfm.database.windows.net
Original connection string copied from the Azure portal:
Server=tcp:sqlserverhcjkjkipgiiulfm.database.windows.net,1433;Initial Catalog=medical;Persist Security Info=False;User ID=Student;Password=AZ500-Lab@10;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

Updated connection string with Always Encrypted enabled:
Data Source=tcp:sqlserverhcjkjkipgiiulfm.database.windows.net,1433;Initial Catalog=medical;Persist Security Info=False;User ID=Student;Password=AZ500-Lab@10;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Column Encryption Setting=Enabled

Enter server password:
```

A red box highlights the "Enter server password:" prompt. On the right, the "Task List" pane shows one task:

- 20. To terminate the console app, press the Enter key

The status bar at the bottom of the Command Prompt window indicates "96% Tasks Complete".

Conclusion

To conclude, we have demonstrated how we can provision cloud infrastructure using an ARM Template. We have also managed to deploy and configure Azure KeyVault using the Azure CloudShell. We have also implemented always always-encrypted feature for sensitive columns in our database.