

# 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 displays the Azure Portal interface. The main window shows the 'Overview' page for the deployment 'Microsoft.Template-20250610023740'. A red box highlights the 'Deployment is in progress' status. Below this, deployment details are listed: Subscription (AZ-500T00-A-CSR-2), Resource group (AZ500LAB10-lod52070586), Start time (6/10/2025, 2:37:37 AM), and Correlation ID (e65b554d-c2a2-4832-802b-1e4cbb2192c4). A table at the bottom is partially visible with columns for Resource, Type, Status, and Operation details.

The right sidebar contains instructions for 'Exercise 2: Configure the Key Vault resource with a key and a secret'. It specifies that resources are in the 'East (US)' region. The tasks to be completed are:

- Task 1: Create and configure a Key Vault
- Task 2: Add a key to the Key Vault
- Task 3: Add a secret to the Key Vault

Task 1 details: 'In this task, you will create an Azure Key Vault resource. You will also configure the Azure Key Vault permissions.' The first step is to 'Open the Cloud Shell by clicking the first icon (next to the search bar) at the top right of the Azure portal. If prompted, select PowerShell and Azure Cloud Shell.' The sidebar also shows '15% tasks Complete' and navigation buttons for 'Previous' and 'End'.

## 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 displays the Azure Cloud Shell environment. On the left, a PowerShell terminal window shows the command to create a new Azure Key Vault. The command is: `New-AzKeyVault -VaultName $kvName -ResourceGroupName 'AZ500LAB10-lod52070586' -Location $location -DisableRbacAuthorization`. Below the command, the output lists the properties of the newly created Key Vault, including its name, resource group, location, and URI.

On the right, a sidebar titled 'Key Vault (Implementing Secure Data by setting up ...)' provides instructions for the task. It includes a 'powershell' section with the command: `'az500kv' + $(Get-Random)`. Below the command, it states: 'The output of the last command will display the vault name and the vault URI. The vault URI is in the format'.

The bottom of the interface shows a progress bar indicating '17% Task Complete' and navigation buttons for 'Previous' and 'End'.

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

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome

labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Microsoft Azure

az500kv1288603018 | Access policies

Key vault

+ Create Refresh Delete Edit

Access policies enable you to have fine grained control over access to vault items. Learn more

Search Permissions: All Type: All

Showing 1 to 1 of 1 records.

Name	Email	Key Permissions	Secret Permissions	Certificate P
LabUser-52070586	LabUser-52070586@clou...	Get, List, Update, Create, ...	Get, List, Set, Delete, Rec...	Get, List, Upi

Task 2: Add a key to Key Vault

This task, you will add a key to the Key Vault and view information about the key.

34% Tasks Complete

< Previous End >

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

```
$kv = Get-AzKeyVault -ResourceGroupName 'AZ500LAB10-lod52070586'  
$key = Add-AzKeyVaultKey -VaultName $kv.VaultName -Name 'MyLabKey' -Destination 'Software'
```

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome

labclient.labondemand.com/LabClient/87b5643b-4f75-4400-a474-66d6e7d4c30b

Microsoft Azure

az500kv1288603018 | Access policies

Key vault

+ Create Refresh Delete Edit

Access policies enable you to have fine grained control over access to vault items. Learn more

Search Permissions: All Type: All

Showing 1 to 1 of 1 records.

Name	Email	Key Permissions	Secret Permissions	Certificate P
LabUser-52070586	LabUser-52070586@clou...	Get, List, Update, Create, ...	Get, List, Set, Delete, Rec...	Get, List, Upi

Task 2: Add a key to Key Vault

This task, you will add a key to the Key Vault and view information about the key.

29% Tasks Complete

< Previous End >

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`

The screenshot displays the Microsoft Azure Cloud Shell environment. The PowerShell session shows the following commands and output:

```
PS /home/labuser-52070586> $secretvalue = ConvertTo-SecureString 'Pa55w.rd1234' -AsPlainText -Force
PS /home/labuser-52070586> $secret = Set-AZKeyVaultSecret -VaultName $kv.VaultName -Name 'SQLPassword' -SecretValue $secretvalue
PS /home/labuser-52070586> Get-AZKeyVaultSecret -VaultName $kv.VaultName
```

The output of the final command is as follows:

```
Vault Name : az500kv1288603018
Name       : SQLPassword
Version    :
Id         : https://az500kv1288603018.vault.azure.net:443/secrets/SQLPassword
Enabled    : True
Expires    :
Not Before :
Created    : 6/10/2025 10:10:01 AM
Updated    : 6/10/2025 10:10:01 AM
Content Type :
Tags       :
```

On the right side of the interface, a task pane titled "Key Vault (Implementing Secure Data by setting up ...)" is visible, showing instructions for creating and verifying the secret. The task pane indicates that 36 minutes remain and shows a progress bar at 35% completion.

# 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 the 'Register an application' blade in the Azure portal. The 'Web' platform is selected, and the redirect URI 'https://sqlapp' is entered. The 'Register' button is highlighted. To the right, a task list shows step 4 completed: 'On the Register an application blade, click Register.'

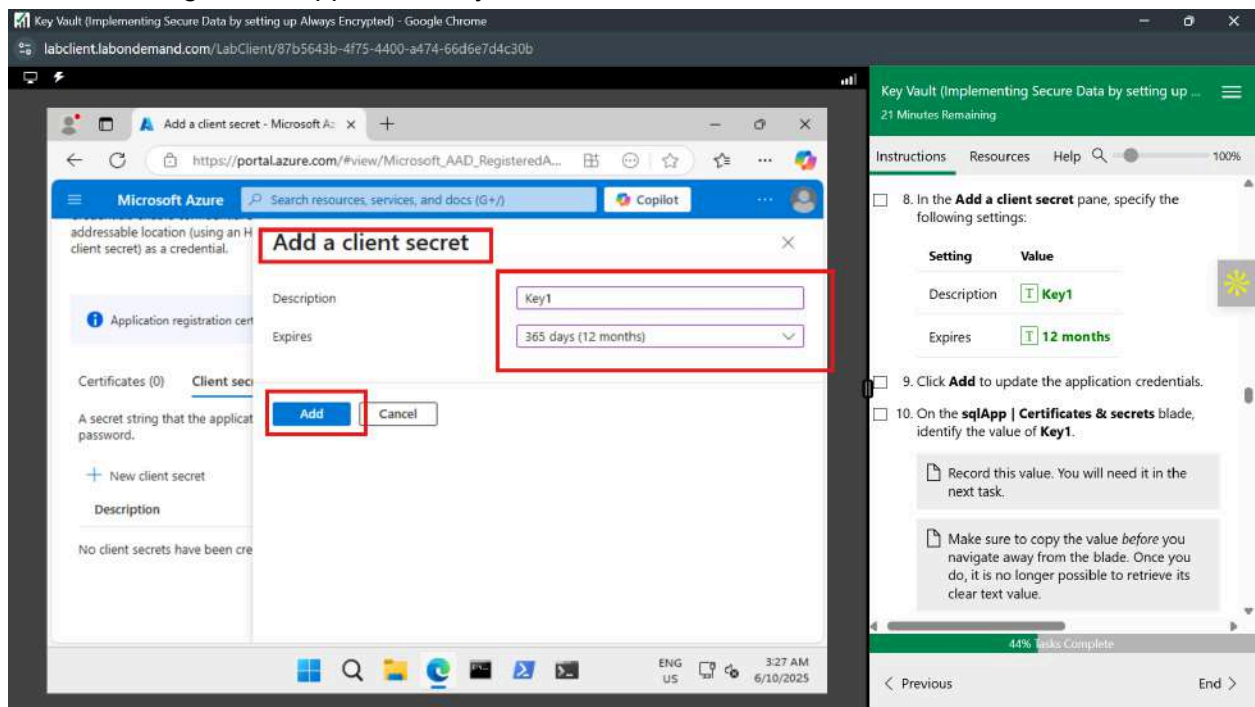
Setting	Value
Name	sqlApp
Redirect URI (optional)	Web and https://sqlapp

- 4. On the **Register an application** blade, click **Register**.
  - Once the registration is completed, the browser will automatically redirect you to **sqlApp** blade.
- 5. On the **sqlApp** blade, identify the value of **Application (client) ID**.
  - Record this value. You will need it in the next task.
- 6. On the **sqlApp** blade, in the **Manage** section,

42% Task Complete



We then configure the app secret key as shown below.



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  
AZ500LAB10-1od52070586 -ServicePrincipalName $applicationId  
-PermissionsToKeys get,wrapKey,unwrapKey,sign,verify,list
```

The screenshot shows a Microsoft Azure Cloud Shell session in a browser window. The terminal displays the command to grant permissions to a Key Vault:

```
PS /home/labuser-52070586> Set-AZKeyVaultAccessPolicy -VaultName $kvName -ResourceGroupName AZ500LAB10-1od52070586 -ServicePrincipalName $applicationId -PermissionsToKeys get,wrapKey,unwrapKey,sign,verify,list
```

The command is highlighted with a red box. To the right, a task instruction panel titled "Key Vault (Implementing Secure Data by setting up Always Encrypted)" shows a progress bar at 54 minutes remaining and a 100% completion status. The panel includes instructions for running the PowerShell command in the Cloud Shell and a task titled "Task 3: Retrieve SQL Azure database ADO.NET Connection String".



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;

The screenshot shows the Microsoft Azure portal interface. On the left, a search bar is visible. The main content area displays the 'ADO.NET (SQL authentication)' section, which is highlighted with a red box. The connection string is as follows:

```
Server=tcp:sqlservrjv2u4kwrmdsg.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;
```

Below the connection string, there is a link to 'Download ADO.NET driver for SQL server'. On the right side of the screenshot, a 'Key Vault' sidebar is visible, showing instructions for setting up Always Encrypted. It includes a task titled 'Task 4: Log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19' and a progress indicator showing '57% Tasks Complete'.

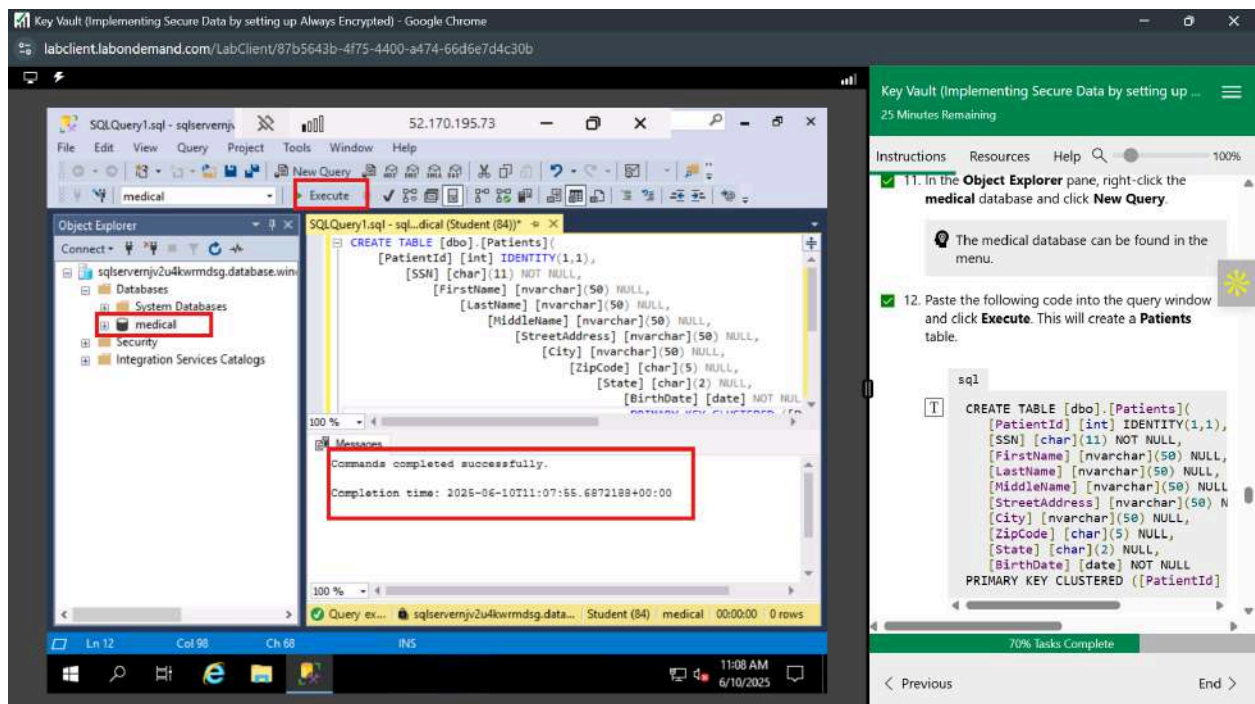
Now we can configure our SQL database firewall rules to allow our virtual machine that was deployed earlier to publicly access it using the following firewall rules.

The screenshot shows the Microsoft Azure portal interface. The main content area displays the 'Firewall rules' section, which is highlighted with a red box. The 'Add a firewall rule' dialog box is open, showing the following details:

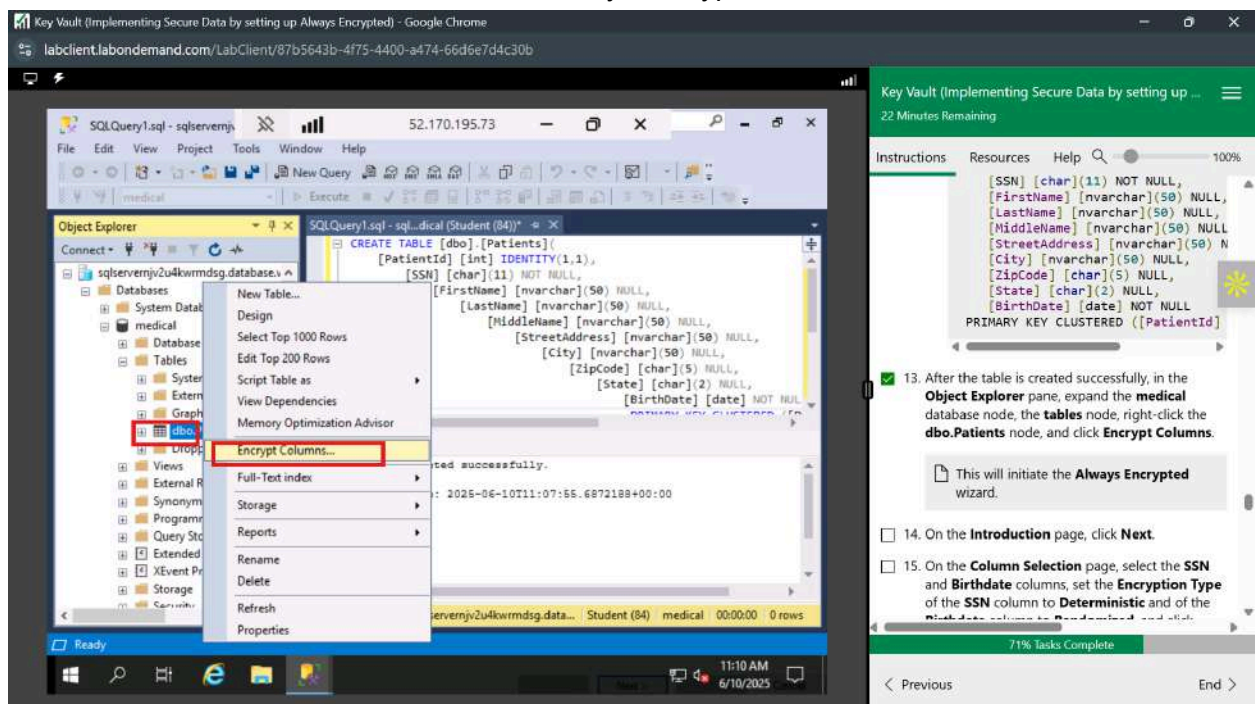
Rule name	Start IP	End IP
Allow Mgmt VM	52.170.195.73	52.170.195.73

Below the dialog box, there are 'Save' and 'Discard' buttons. On the right side of the screenshot, a 'Key Vault' sidebar is visible, showing instructions for setting up Always Encrypted. It includes a task titled 'Task 4: Log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19' and a progress indicator showing '60% Tasks Complete'.

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.Patie...			
Date			
<input checked="" type="checkbox"/> SSN	⚠	Deterministic	CEK_Auto1 (New)
FirstN...			
LastN...			
Middl...			
Street...			
City			
ZipC...			
State			
<input checked="" type="checkbox"/> Birth...		Randomized	CEK_Auto1 (New)

☐ Show affected columns only

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.S...** 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

72% Tasks Complete

< Previous End >

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

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

< Previous End >



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

The image consists of two screenshots from a lab environment, showing the completion of an Always Encrypted setup in SQL Server.

**Top Screenshot:** The 'Always Encrypted Wizard Log Report' window is open, displaying a summary of tasks. A red box highlights the following tasks and their status:

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

The 'Results' pane on the left shows the 'Summary' section selected. The 'Package Manager Console' on the right shows the installation of NuGet packages: Microsoft.SqlServer and Microsoft.IdentityModel.

**Bottom Screenshot:** The 'SQL Server Enterprise Manager' window is open, showing the 'Object Explorer' pane. The 'Always Encrypted Keys' subnode is expanded, showing 'Column Master Keys' and 'Column Encryption Keys'. The 'Messages' pane shows the completion of the encryption process:

```
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
)
GO
```

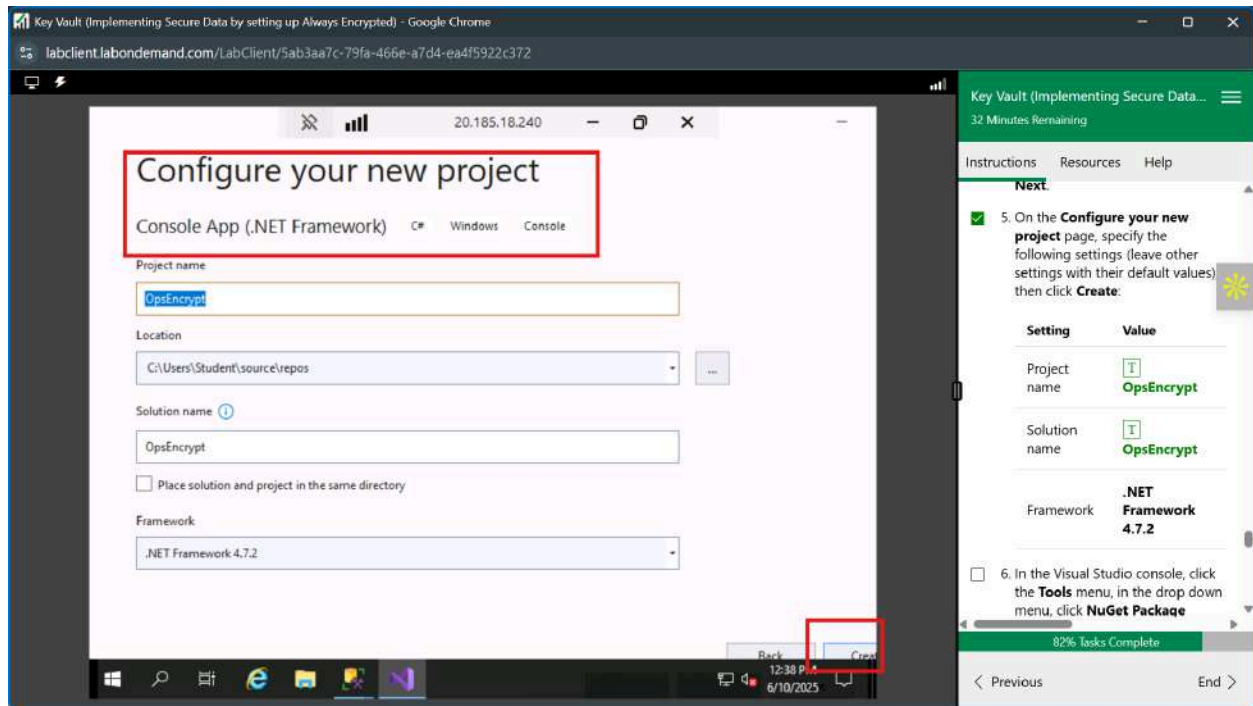
The 'Messages' pane shows the completion of the encryption process:

```
Commands completed successfully.
Completion time: 2025-06-10T11:07:55.6872188+00:00
```

The 'Package Manager Console' on the right shows the installation of NuGet packages: Microsoft.SqlServer and Microsoft.IdentityModel.

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

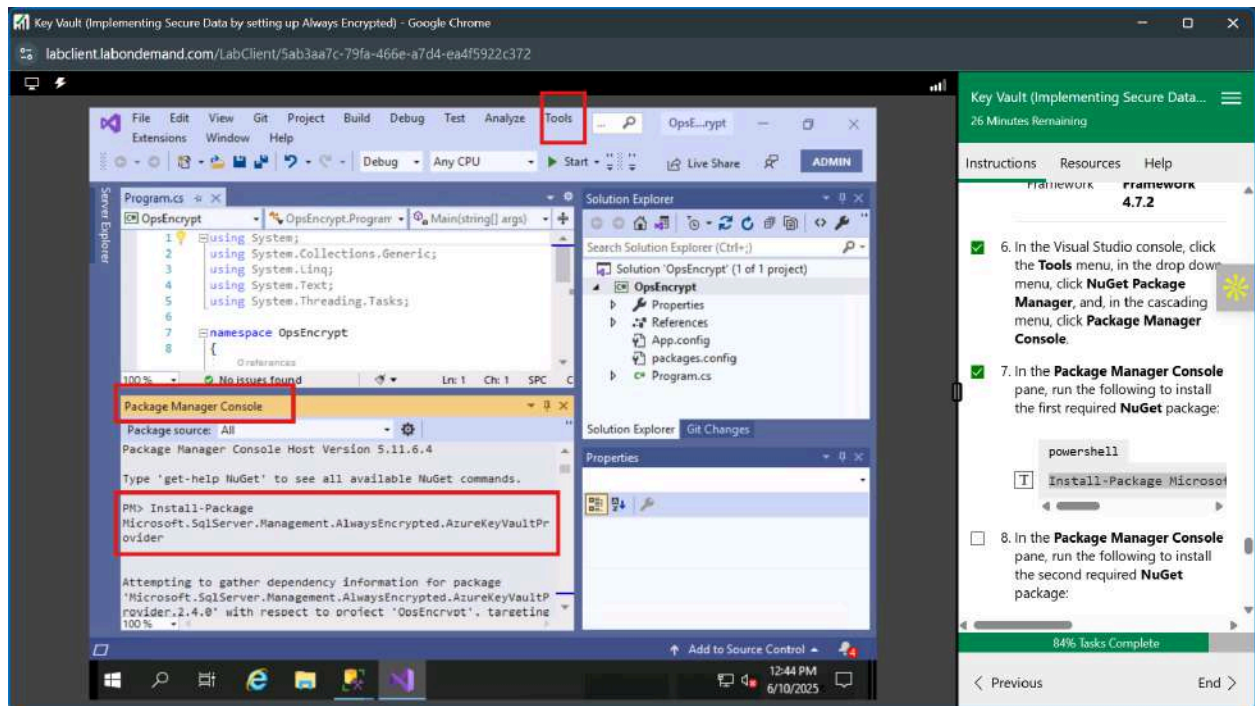
In this task, we will run a data-driven application to demonstrate the use of Azure Key Vault in encrypting the Azure SQL database. We will create a console application using Visual Studio to load data into the encrypted columns and then access that data securely using a connection string that accesses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



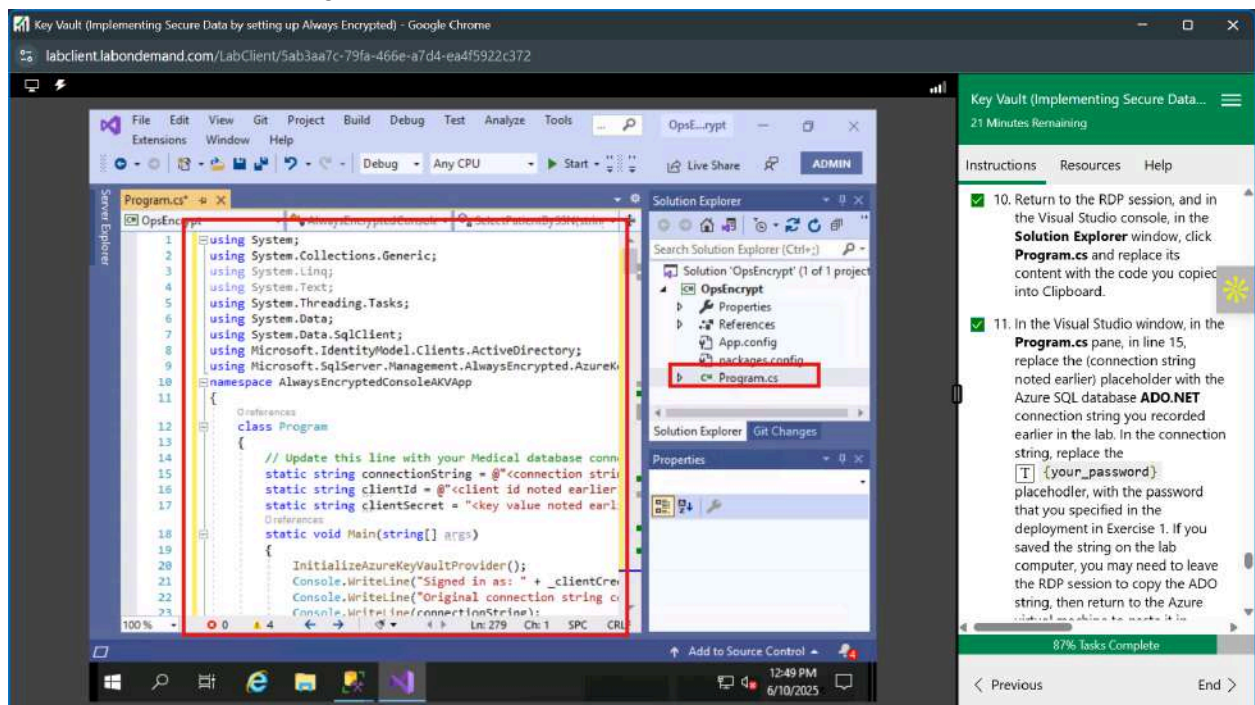
We then navigate to the tools menu and install necessary packages using the following command;

`Install-Package`

`Microsoft.SqlServer.Management.AlwaysEncrypted.AzureKeyVaultProvider`

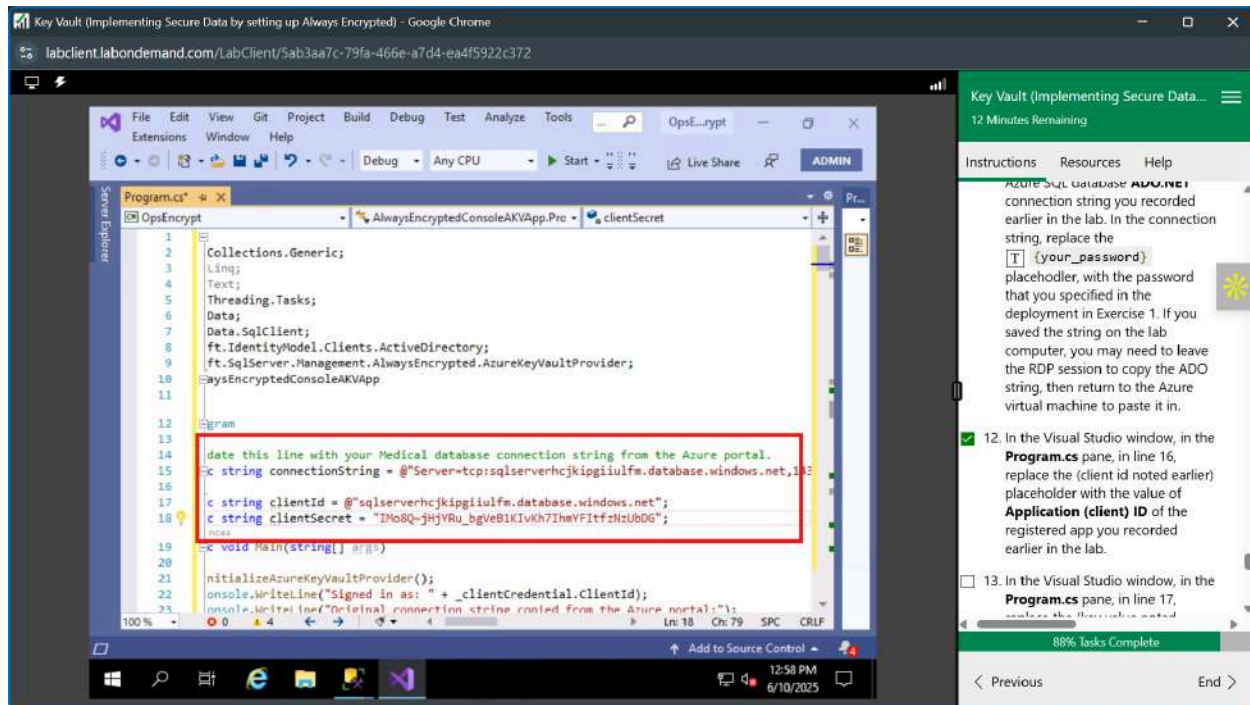


We then paste the C program code in our lab files into the new Visual Studio window.





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;

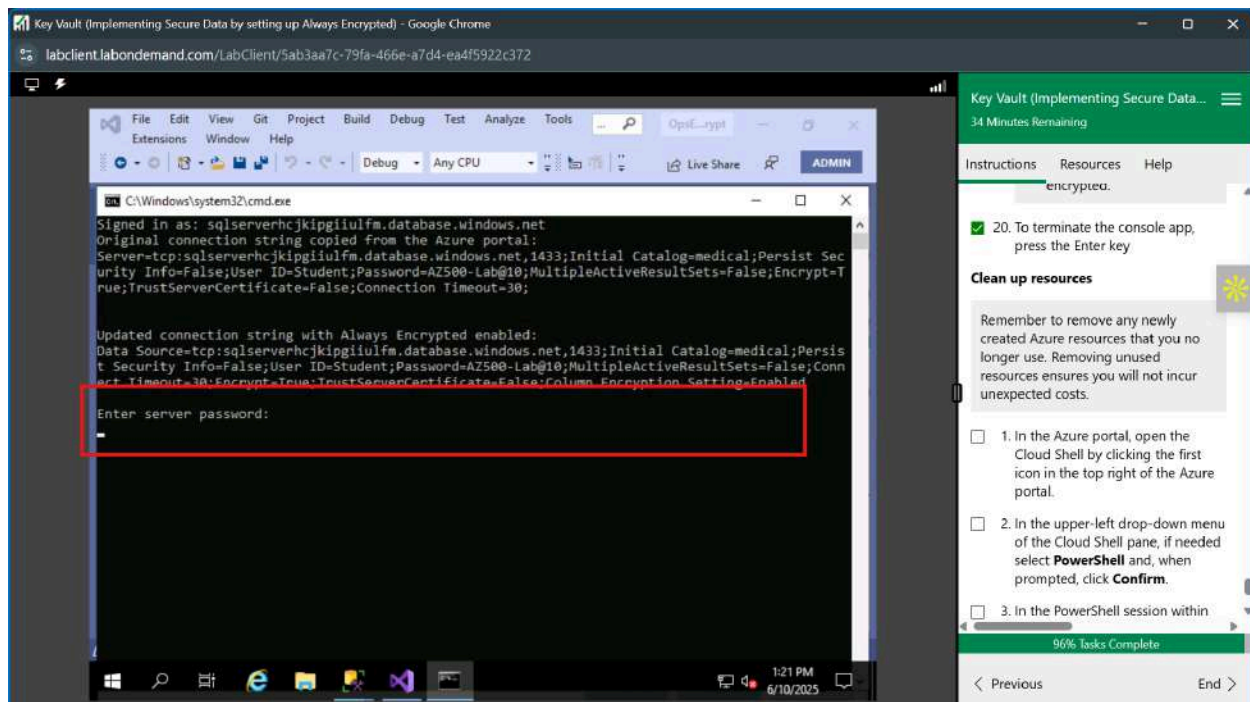


The screenshot shows the Visual Studio Code editor with the file `Program.cs` open. A red box highlights the following code lines:

```
15: string connectionString = @"Server=tcp:sqlserverhckipgiulfm.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;";
16: string clientId = @"sqlserverhckipgiulfm.database.windows.net";
17: string clientSecret = "IMo8Q-jHjVRU_gVe8K1vkh7IheVFtFzHzUb0G";
```

The right sidebar shows the 'Key Vault (Implementing Secure Data...)' task pane with instructions for replacing placeholders in the connection string and client ID.

We then run the code as shown below



The screenshot shows a terminal window with the following output:

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

Enter server password:
```

The right sidebar shows the 'Key Vault (Implementing Secure Data...)' task pane with instructions for terminating the console app and cleaning up resources.

# 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.