ASSIGNMENT TWO: KEY VAULT AND ENCRYPTION WRITEUP

BY CS-CNS03-23082 – ABUOR ISABELLA MERCY

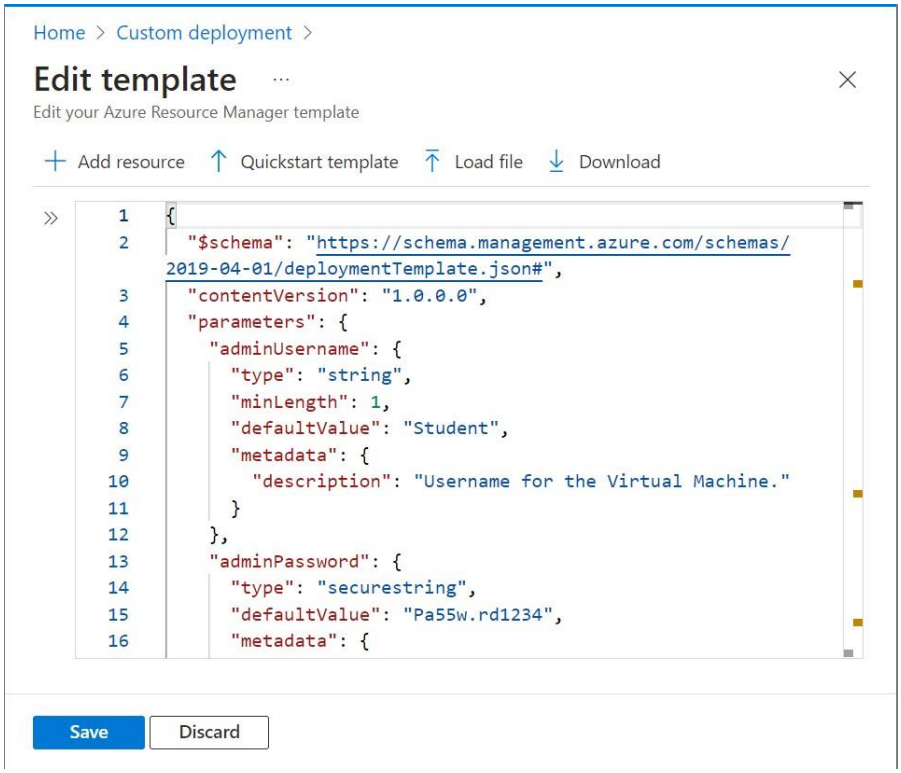
INTRODUCTION

This lab is divided into four exercises, One deploying the base infrastructure from an ARM template, the second exercise comprises of a series of tasks to configure the Key Vault resource with a key and a secret, the third exercise focuses on configuring an Azure SQL database and a data-driven application and the final exercise focuses on demonstrating the use of Azure Key Vault in encrypting the Azure SQL database. The following write-up entails a step by step process on how the following exercises were completed in order to implement Secure Data

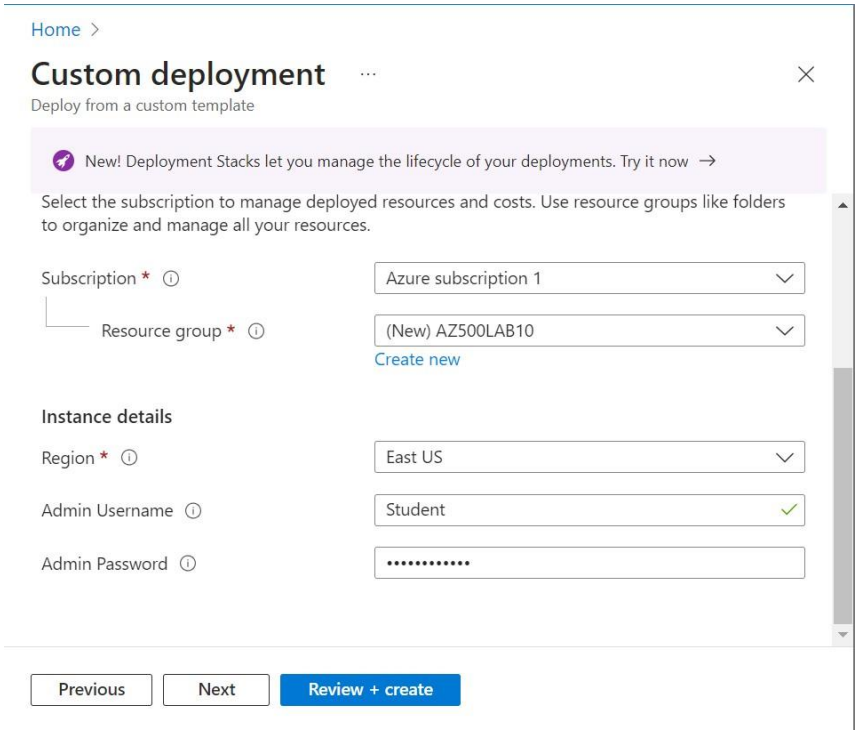
Exercise 1: Deploy the base infrastructure from an ARM template

This exercise consisted of a step by step process of deploying an Azure VM and an Azure SQL database which will automatically install Visual Studio 2019 and SQL Server Management Studio 19 as part of the deployment

In deploy a custom template, on the custom deployment blade, we edited template blade by uploading file the az-500-10_azuredeploy.json file.



Then under **Deployment Scope** the following settings are configured and the created.



Exercise 2: Configure the Key Vault resource with a key and a secret

This exercise involved a series of tasks which were:

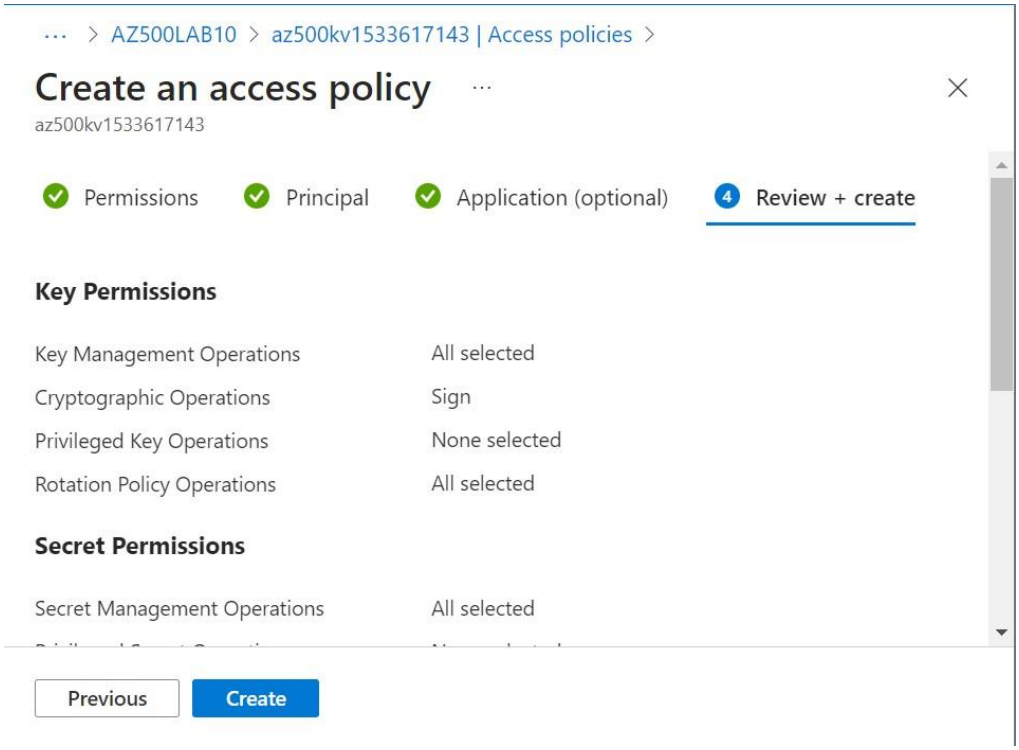
Task 1: Create and configure a Key Vault

In this task, we created an Azure Key Vault resource and configured the Azure Key Vault permissions.

On the Cloud Shell the following code was used to create an Azure Key Vault in the resource group AZ500LAB10.

```
PS /home/mary>  $kvName = 'az500kv' + $(Get-Random)
PS /home/mary>
PS /home/mary>  $location = (Get-AzResourceGroup -ResourceGroupName 'AZ
500LAB10').Location
PS /home/mary>
PS /home/mary>  New-AzKeyVault -VaultName $kvName -ResourceGroupName 'A
Z500LAB10' -Location $location
```

On the Resource group blade, in the list of resource group, on AZ500LAB10 entry, we clicked on the newly created Key Vault, under Access policies, in the Overview section. We created an access policy and specify the following settings



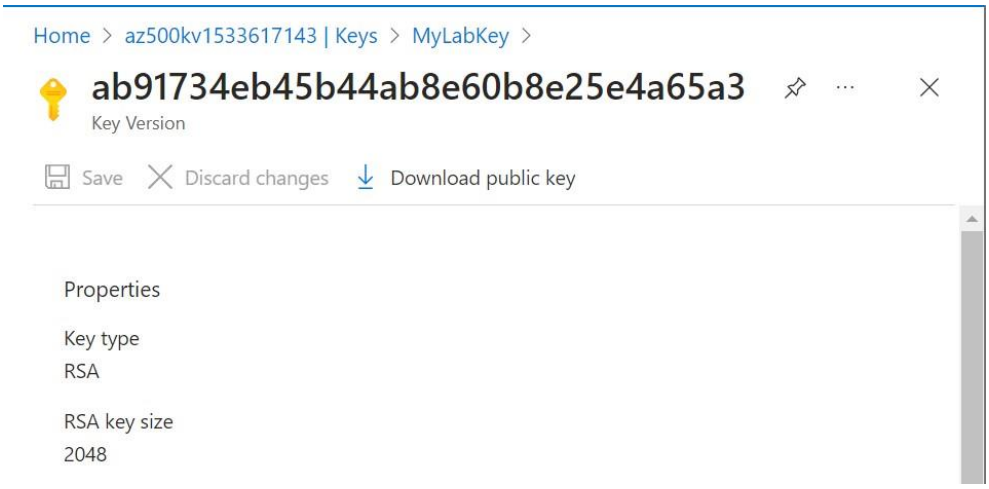
Task 2: Add a key to Key Vault

In this task, we added a key to the Key Vault and viewed information about the key.On the Cloud Shell pane, the following code was prompted to add a software-protected key to the Key Vault and verify if the key was created.

```
PS /home/mary>  $kv = Get-AzKeyVault -ResourceGroupName 'AZ500LAB10'
PS /home/mary>
PS /home/mary>  $key = Add-AZKeyVaultKey -VaultName $kv.VaultName -Name
 'MyLabKey' -Destination 'Software'
PS /home/mary>  Get-AZKeyVaultKey -VaultName $kv.VaultName
```

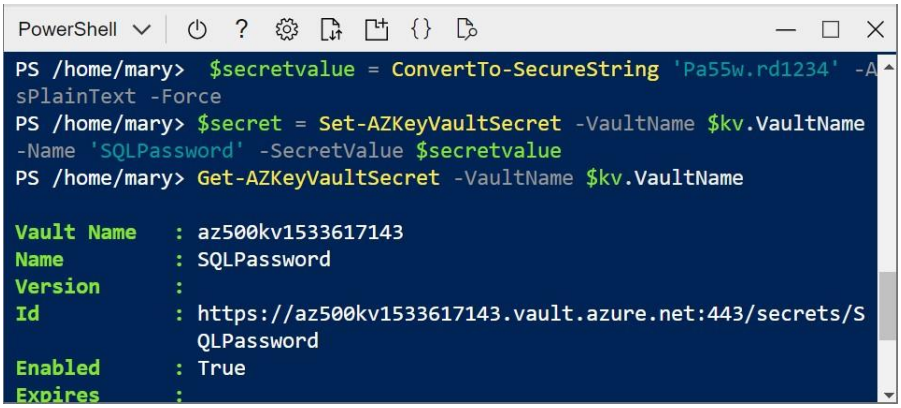
We then prompted the following to display the key identifier:

```
PS /home/mary>  $key.key.kid
https://az500kv1533617143.vault.azure.net/keys/MyLabKey/ab91734eb45b44a
b8e60b8e25e4a65a3
PS /home/mary>
```

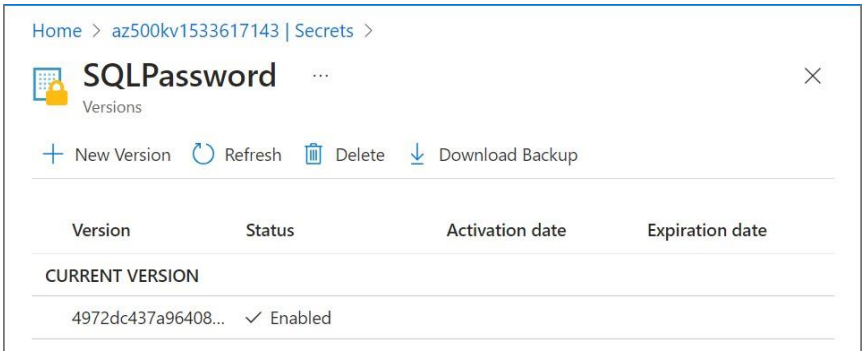


Task 3: Add a Secret to Key Vault

On the Cloud Shell pane, the following was used to create a variable with a secure string value, then second to add the secret to the vault and lastly to verify the secret was created.



We then navigated back to the Key Vault blade, in the Objects section, under Secrets to verify the information of the secret that was created.



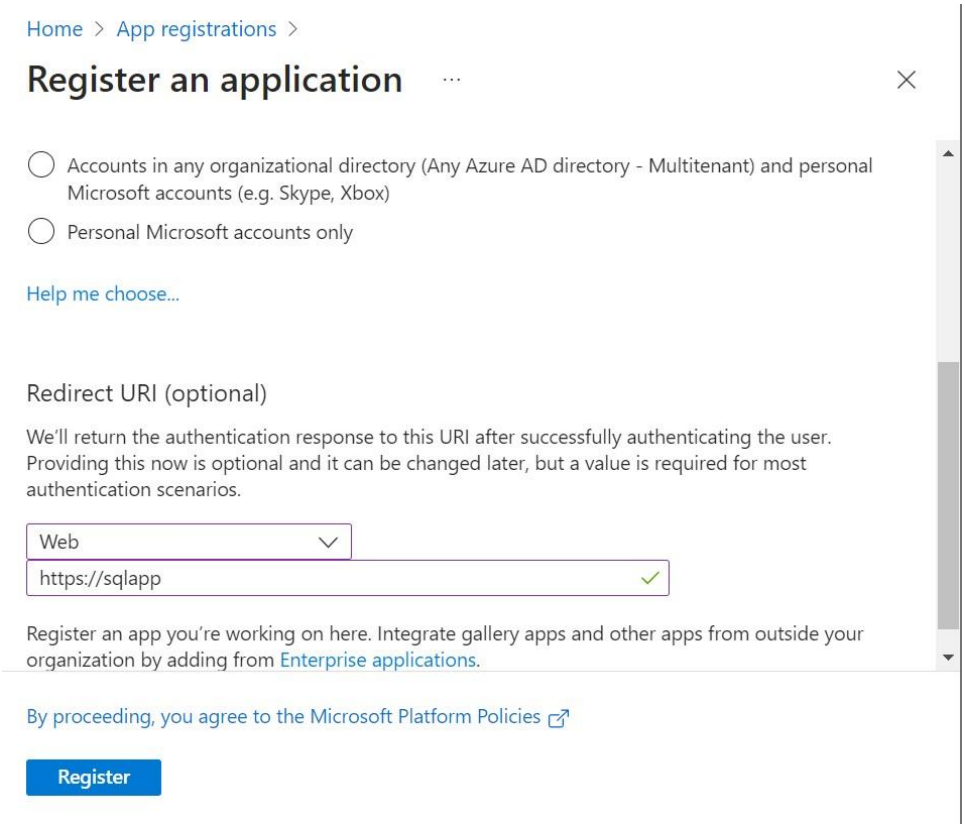
Exercise 3: Configure an Azure SQL database and a data-driven application

This exercise consisted of five tasks to configure an SQL database. The detailed tasks are as follows:

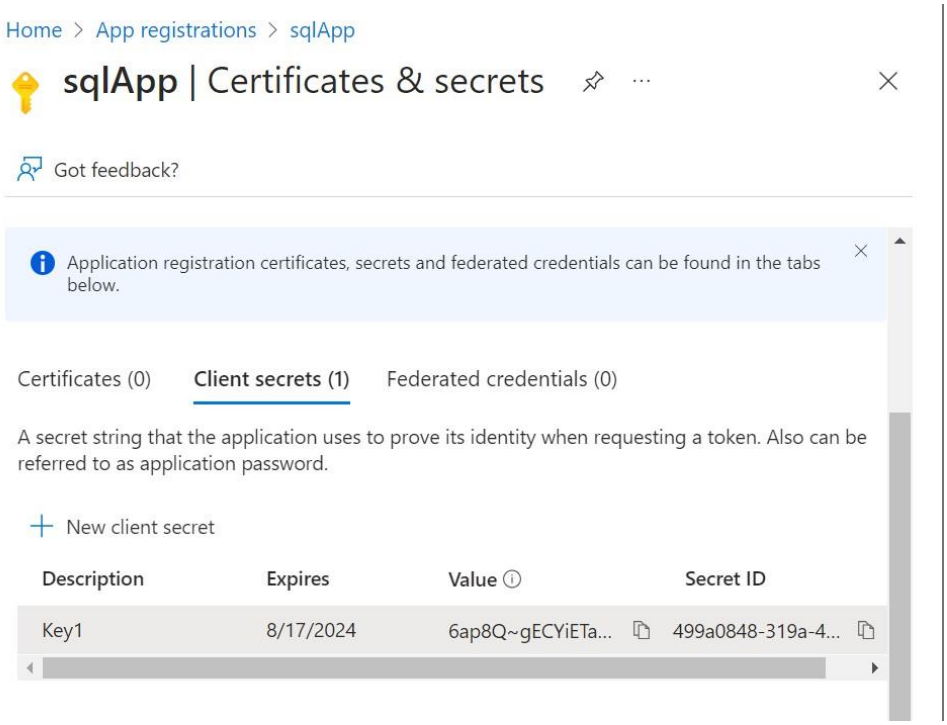
Task 1: Enable a client application to access the Azure SQL Database service.

In this task, we enabled a client application to access the Azure SQL Database service.

In the Azure portal, in the App Registrations blade we created a new registration with the following settings:



On the sqlApp blade, in the Manage section, on Certificates & secrets we created a new client secret and specified the following settings:



Task 2: Create a policy allowing the application access to the Key Vault.

In this task, we will grant the newly registered app permissions to access secrets stored in the Key Vault.

On the Cloud Shell pane, the following was used to create a variable storing the Application (client) ID that was recorded in the previous task. Then a variable was created to store the Key Vault name, then finally to grant permissions on the Key Vault to the application that was registered in the previous task:

```
PS /home/mary>  $applicationId = '9c8e5467-efea-4ad2-9aff-2ef07e0ce91c'

PS /home/mary>  $kvName = (Get-AzKeyVault -ResourceGroupName 'AZ500LAB1
0').VaultName
PS /home/mary>
PS /home/mary>  $kvName
az500kv1533617143
PS /home/mary>  Set-AZKeyVaultAccessPolicy -VaultName $kvName -Resource
GroupName AZ500LAB10 -ServicePrincipalName $applicationId -PermissionsT
oKeys get,wrapKey,unwrapKey,sign,verify,list
```

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. We updated the empty database resource with a new table structure and select data columns for encryption

On SQL databases in the list of SQL databases, under the **medical()** entry, in the Settings section, under Connection strings we recorded ADO.NET (SQL authentication) connection string..

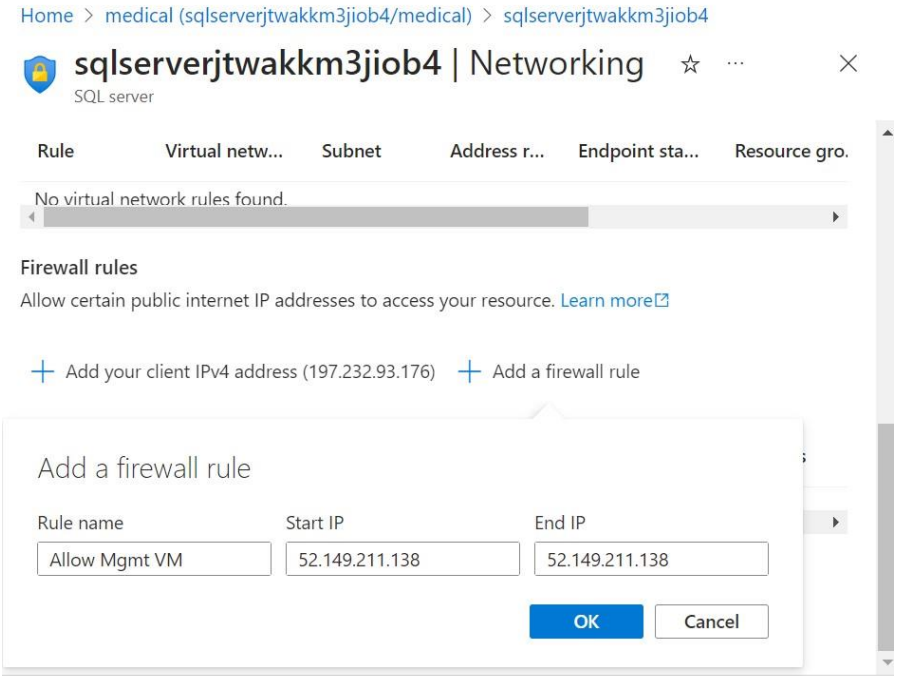
Task 4: Log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19

In this task, we log on to the Azure VM, which deployment you initiated in Exercise 1. On virtual machines, In az500-10-vm1 entry. On the az500-10-vm1 blade, we took note of the Public IP address.

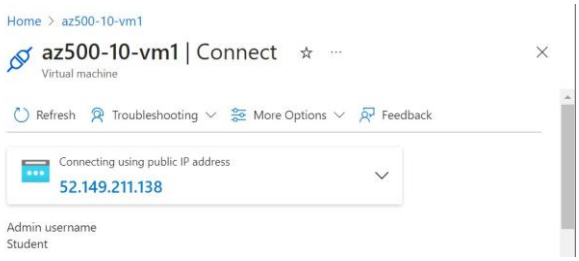
Task 5: Create a table in the SQL Database and select data columns for encryption

In this task, we connected to the SQL Database with SQL Server Management Studio and create a table. we then encrypted two data columns using an autogenerated key from the Azure Key Vault.

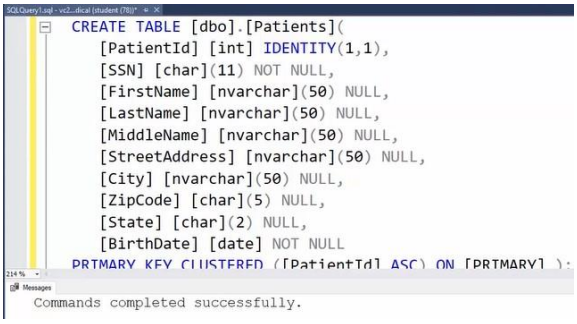
In the medical SQL database, in the Essentials section, identify the Server name (copy to clipboard), and then, in the toolbar, we set server firewall and Added a firewall rule, with the following settings:



We then navigated back to the az500-10-vm1 blade, and Connect and, using RDP and Downloaded the RDP File to use it to connect to the az500-10-vm1 Azure VM via Remote Desktop.



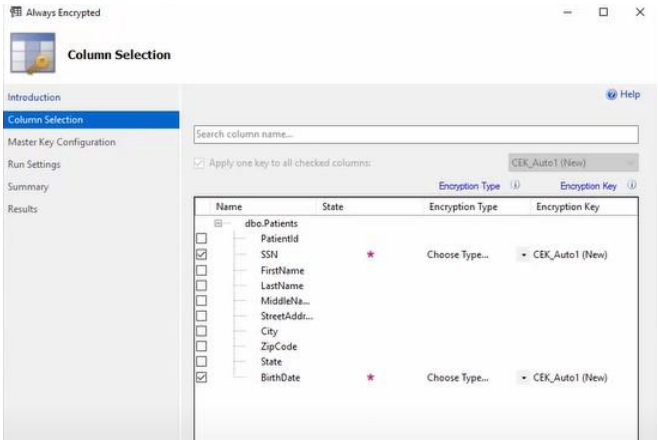
We then In the Connected to Server, Within the SQL Server Management Studio console, in the Object Explorer pane, we created a new Query and posted the following code to create a patients table.



```
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
PRIMARY KEY CLUSTERED ([PatientId] ASC) ON [PRIMARY] ):
```

```
Commands completed successfully.
```

After the table is created successfully, we encrypted the columns in the dbo.Patients node. On the Column Selection page, we selected the SSN and Birthdate columns, set the Encryption Type of the SSN column to Deterministic and of the Birthdate column to Randomized.

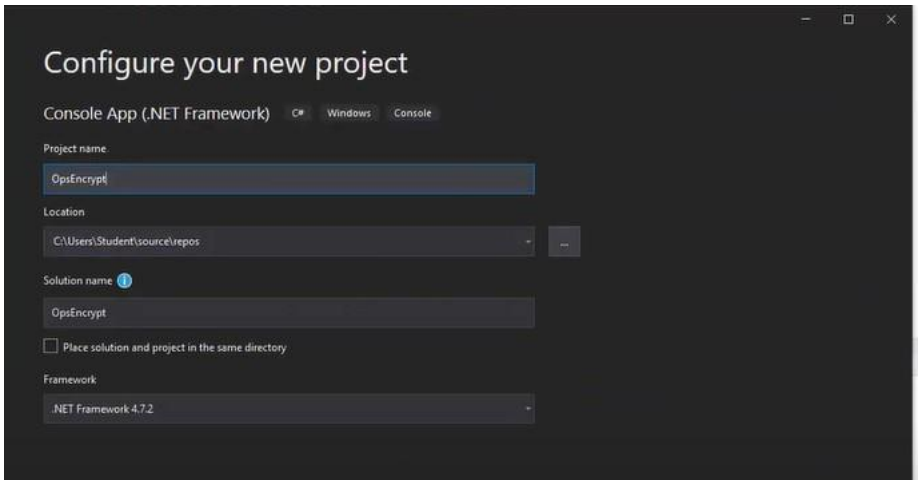


Exercise 4: Demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

In this exercise, we ran a data-driven application to demonstrate the use of Azure Key Vault in encrypting the Azure SQL database.

We created 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.

From visual studio 2019 we create a new project with Console App (.NET Framework) for C# template and configured the project as shown below:



In the Visual Studio console, on the Tools menu, in the drop down menu, clicked NuGet Package Manager, and, in the cascading menu, click Package Manager Console.

We then installed NuGet packages.We then clicked **Program.cs** and replace its connection string, password, client id and key value noted earlier in the assignment.

```
// Update this line with your Medical database connection string from the Azure portal.
static string connectionString = @"Server=tcp:vc20200923.database.windows.net,1433;Initial Catalog=medical;Per
static string clientId = @"3e5e54b8-3aa0-4127-8f83-90927ff5bbef";
static string clientSecret = "SRu28C601y1_8JWqYxNdIXX1-tqgS~t5~H";
0 references
```

Then clicked start to initiate the build of the console application and start it.

```
igned in as: 3e5e54b8-3aa0-4127-8f83-90927ff5bbef
riginal connection string copied from the Azure portal:
erver=tcp:vc20200923.database.windows.net,1433;Initial Catalog=medical;Persist Security Info=False;User ID=Student;Pass
ord=Pa55w.rd1234;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

pdated connection string with Always Encrypted enabled:
ata Source=tcp:vc20200923.database.windows.net,1433;Initial Catalog=medical;Persist Security Info=False;User ID=Student
Password=Pa55w.rd1234;MultipleActiveResultSets=False;Connect Timeout=30;Encrypt=True;TrustServerCertificate=False;Colum
Encryption Setting=Enabled

nter server password:
a55w.rd1234

dding sample patient data to the database...
ll the records currently in the Patients table:
rlando Gee     SSN: 999-99-0001        Birthdate: 1/4/1964 12:00:00 AM
eith Harris    SSN: 999-99-0002        Birthdate: 6/20/1977 12:00:00 AM
onna Carreras  SSN: 999-99-0003        Birthdate: 2/9/1973 12:00:00 AM
anet Gates     SSN: 999-99-0004        Birthdate: 8/31/1985 12:00:00 AM
ucy Harrington SSN: 999-99-0005        Birthdate: 5/6/1993 12:00:00 AM

ow lets locate records by searching the encrypted SSN column.
lease enter a valid SSN (ex. 999-99-0003):
99-99-0003
atient found with SSN = 999-99-0003
onna Carreras  SSN: 999-99-0003        Birthdate: 2/9/1973 12:00:00 AM
ress Enter to exit...
```

In the SQL Management Studio console, in the Object Explorer pane, on the medical database and, created a new Query

```
SELECT FirstName, LastName, SSN, BirthDate FROM Patients;
```

and then the following query to verify that the data that loaded into the database from the console app is encrypted.

Results | Messages

	FirstName	LastName	SSN	BirthDate
1	Orlando	Gee	0x01C27519ED25CEE5898A5124CC4DE6F9EAF17EC1E5F1418...	0x010CF941E5DF634EEA225443C1485A4DF8179713D6C5D59...
2	Keith	Harris	0x01AC2BB18C1F40B499009971DF24532127563C2ADF712CE...	0x011F56F1E8A848F8FDCF430BC1E04596DBC01A5B827E3FF...
3	Donna	Carreras	0x01E987649C2315C630C842D61786D0DDB36D461EB28D3A...	0x0105827BD61AEDD1A4B9642020F460F9E7CAF98789170E3...
4	Janet	Gates	0x0199A5C115D839674D4A312CF989795F55277E9278E2EA5F...	0x01FBE7403687EE4D76FD791D6962D5F16D0C468CF448F0E...
5	Lucy	Harrington	0x011FB6C464A8C6201E49E25AA1541F31DDA6238EE2A89CC...	0x013EC81CCDEF21608AEA5C4445E85D27625B763B3FDE68...

CONCLUSION

In conclusion, through the completion of each exercise, the main goals were to create a safe place to store important information and make sure that a database's contents are super secure in which was a success. Firstly, we built an Azure Key Vault where the secret codes and valuable keys are kept. Not to mention learning about the encryption type through the key type and size.

Next, we tackled a SQL Database. Learning about the smart method called Always Encrypted to lock up certain parts of the database. By accomplishing these tasks, I have gained the experience in managing sensitive information, and orchestrating data-driven applications.