

Week 9 Assignment : Key Vault And Encryption (Implementing Secure Data by setting up Always Encrypted)

Introduction

Key Vault is a service for securely managing cryptographic keys and secrets, while encryption is the process of scrambling data to protect it. Key Vault protects data by generating and storing encryption keys and then using them to encrypt and decrypt data as needed, ensuring the keys themselves are never exposed. **Always Encrypted** is a specific technology that is designed to protect sensitive data by ensuring it is encrypted even while it is being used in an application or database. This technology encrypts sensitive data within the client application, so it remains encrypted **both in transit and at rest** on the database, and **only the authorized client applications with the necessary encryption keys can access the unencrypted version**. This integration is used in applications, databases, and cloud services to **secure sensitive information like connection strings and user data**.

Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets. Additionally, Azure Key Vault lets one easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and their internal connected resources.

Azure Key Vault offers two service tiers to meet different security and compliance requirements. In the standard tier, data is encrypted using software libraries validated to FIPS 140 Level 1. The premium tier offers Hardware Security Module (HSM) -protected keys, generated and protected by **FIPS 140-3 Level 3 validated Marvell LiquidSecurity HSMs**, for the highest level of cryptographic protection

Lab scenario

I was asked to create a proof of concept application that makes use of the Azure SQL Database support for Always Encrypted functionality. All of the secrets and keys used in this scenario should be stored in Key Vault. The application should be registered in Microsoft Entra ID in order to enhance its security posture. To accomplish these objectives, the proof of concept should include:

- Creating an Azure Key Vault and storing keys and secrets in the vault.

- Create a SQL Database and encrypting content of columns in database tables by using Always Encrypted.

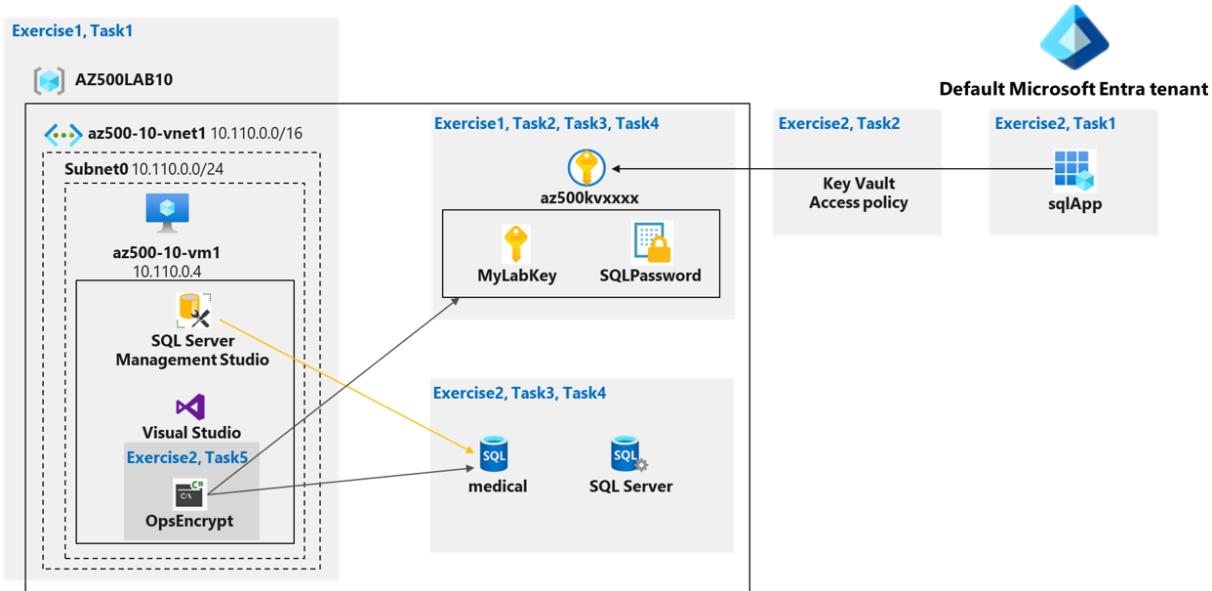
To keep the focus on the security aspects of Azure, related to building this proof of concept, I started from an automated ARM template deployment, setting up a Virtual Machine with Visual Studio 2019 and SQL Server Management Studio 19.

Lab objectives

In this lab, I completed the following exercises:

- Exercise 1: Deploy the base infrastructure from an ARM template
- Exercise 2: Configure the Key Vault resource with a key and a secret
- Exercise 3: Configure an Azure SQL database and a data-driven application
- Exercise 4: Demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

Key Vault diagram



Instructions

Lab files:

- \Allfiles\Labs\10\az-500-10_azuredeploy.json
- \Allfiles\Labs\10\program.cs

Exercise 1: Deploy the base infrastructure from an ARM template

Task 1: Deploy an Azure VM and an Azure SQL database

In this task, I deployed an Azure VM, which I automatically installed Visual Studio 2019 and SQL Server Management Studio 19 as part of the deployment.

1. Sign-in to the Azure portal <https://portal.azure.com/>.
Sign in to the Azure portal using an account that has the Owner or Contributor role in the Azure subscription you are using for this lab. In this **Cloudslice** lab, this account is *LabUser-57171145@cloudslice.onmicrosoft.com with TAP #h%9am%a*.
2. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Deploy a custom template** and press the **Enter** key.
3. On the **Custom deployment** blade, click the **Build your own template in the editor** option.
4. On the **Edit template** blade, click **Load file**, locate the **\Allfiles\Labs\10\az-500-10_azuredeploy.json** file and click **Open**.
5. On the **Edit template** blade, click **Save**.
6. On the **Custom deployment** blade, under **Deployment Scope** ensure that the following settings are configured (leave any others with their default values):

Setting	Value
---------	-------

Subscription the name of the Azure subscription you will be using in this lab

Resource group Use the existing **Resource Group**
AZ500Lab10-1od57171145

Location East US

Username Student

Password Please use your personal password created in Lab 02 > Exercise 2 > Task 1 > Step 3.

7. While you can change the administrative credentials used for logging on to the Virtual Machine, you don't have to.

To identify Azure regions where you can provision Azure VMs, refer to
<https://azure.microsoft.com/en-us/regions/offers/>

8. Click the **Review and Create** button, and confirm the deployment by clicking the **Create** button.

This initiates the deployment of the Azure VM and Azure SQL Database required for this lab.

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/a98ad28d-fd57-4906-8ed7-1067fe62a084

Deploy from a custom template

Microsoft assumes no responsibility for any actions performed by third-party templates and does not provide rights for third-party products or services. See the [Azure Marketplace Terms](#) for additional terms.

Deploying this template will create one or more Azure resources or Marketplace offerings. You acknowledge that you are responsible for reviewing the applicable pricing and legal terms associated with all resources and offerings deployed as part of this template. Prices and associated legal terms for any Marketplace offerings can be found in the [Azure Marketplace](#); both are subject to change at any time prior to deployment.

Neither subscription credits nor monetary commitment funds may be used to purchase non-Microsoft offerings. These purchases are billed separately.

If any Microsoft products are included in a Marketplace offering (e.g. Windows Server or SQL Server), such products are licensed by Microsoft and not by any third party.

Basics

Subscription	AZ-500T00-A CSR 1
Resource group	AZ500LAB10-lod57173201
Region	East US
Admin Username	Student
Admin Password	*****

Previous Next Create

ENG US

To identify Azure regions where you can provision Azure VMs, refer to <https://azure.microsoft.com/en-us/regions/offers/>

7. Click the **Review and Create** button, and confirm the deployment by clicking the **Create** button.

This initiates the deployment of the Azure VM and Azure SQL Database required for this lab.

Do not wait for the ARM template deployment to be completed, but

1 Hr 21 Min Remaining

Key Vault (Implementing Secure Data by setting up Always Encrypted) - Google Chrome
labclient.labondemand.com/LabClient/a98ad28d-fd57-4906-8ed7-1067fe62a084

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

Home > Microsoft.Template-20251130021130 | Overview

Deployment is in progress

Deployment name : Microsoft.Template-20251130021130
Subscription : AZ-500T00-A CSR 1
Resource group : AZ500LAB10-lod57173201
Start time : 11/30/2025, 2:11:28 AM
Correlation ID : 22e31c3d-3b2f-4b42-9286-eabc3c3afb34

Deployment details

Resource	Type	Status
vhdstoragekug...	Storage account	Accepted
az500-10-vnet1	Virtual network	OK
az500-10-pip1	Public IP address	OK
az500-10-nsg1	Network security group	OK

Database required for this lab.

Do not wait for the ARM template deployment to be completed, but instead continue to the next exercise. The deployment might take between 20-25 minutes.

Install the az500-10-DB.json Custom Template

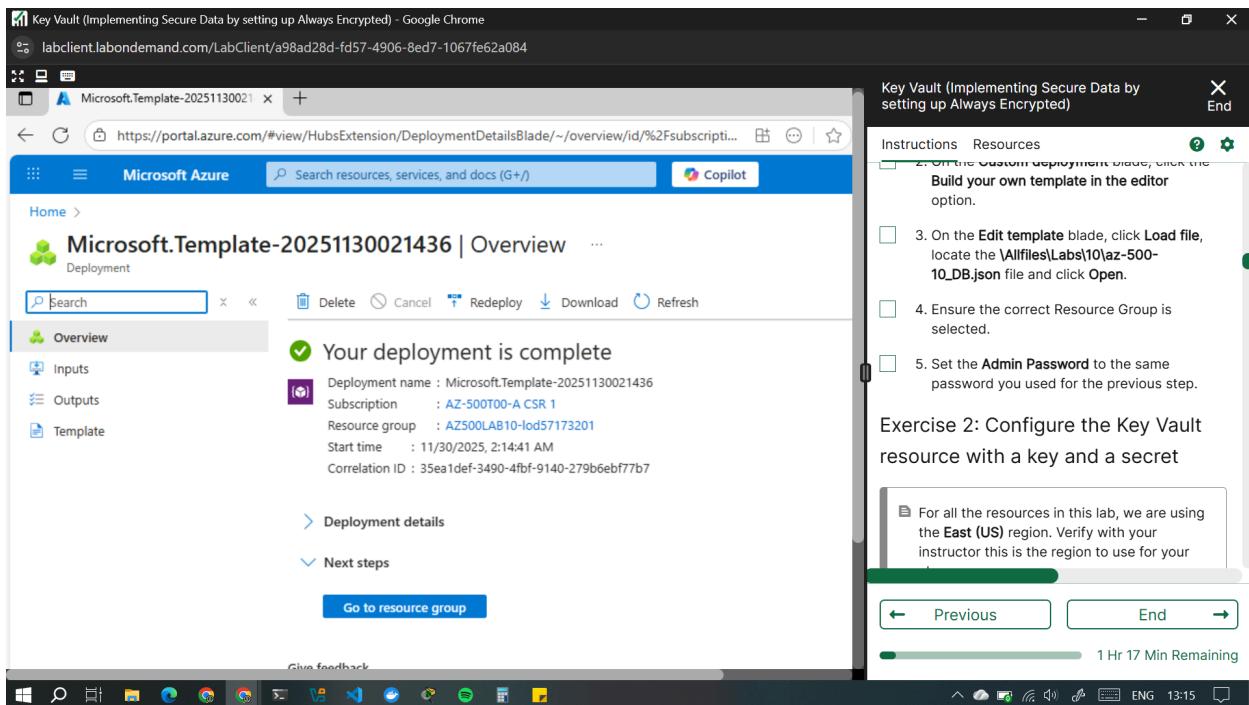
- In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Deploy a custom template** and press the **Enter** key.
- On the **Custom deployment** blade, click the **Build** button.

1 Hr 21 Min Remaining

The deployment took around **20-25 minutes**.

Install the az500-10-DB.json Custom Template

1. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Deploy a custom template** and press the **Enter** key.
2. On the **Custom deployment** blade, click the **Build your own template in the editor** option.
3. On the **Edit template** blade, click **Load file**, locate the **\Allfiles\Labs\10\az-500-10_DB.json** file and click **Open**.
4. Ensure the correct Resource Group is selected.
5. Set the **Admin Password** to the same password you used for the previous step.



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

In this exercise, I completed the following tasks:

-
- 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: Create and configure a Key Vault

In this task, you will create an Azure Key Vault resource. You will also configure the Azure Key Vault permissions.

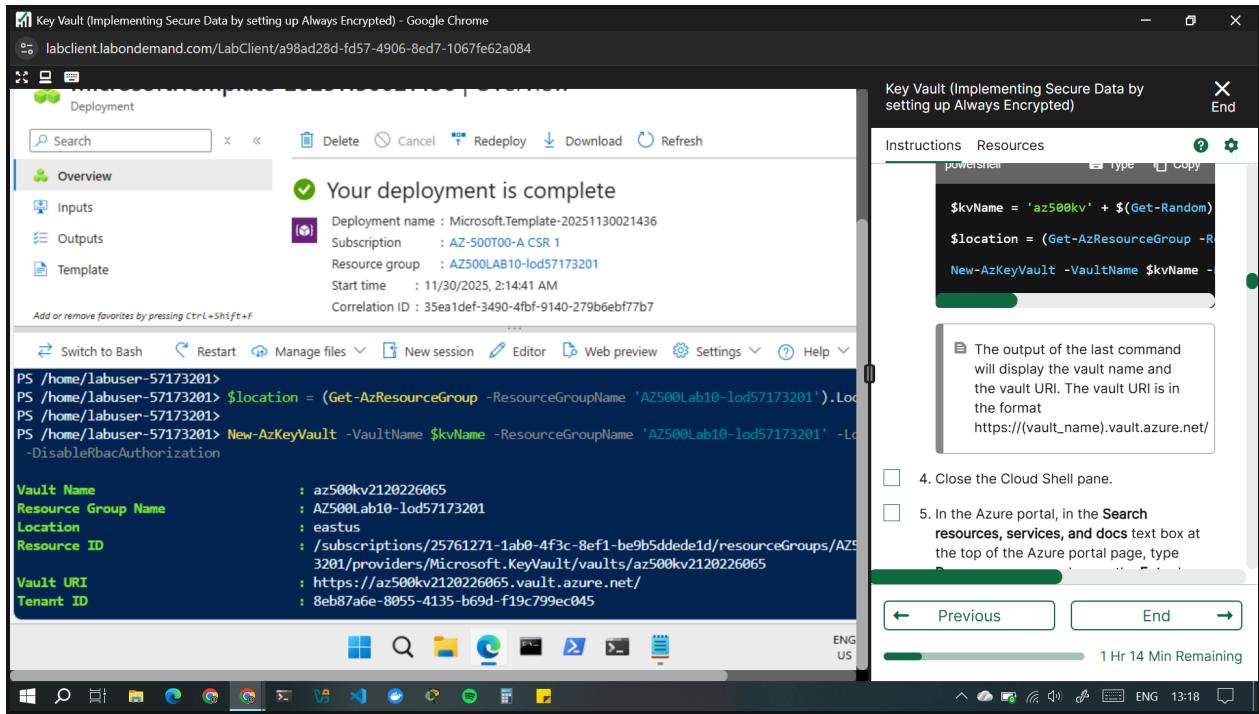
1. Open the Cloud Shell by clicking the first icon (next to the search bar) at the top right of the Azure portal. If prompted, select **PowerShell** and **Create storage**.
2. Ensure **PowerShell** is selected in the drop-down menu in the upper-left corner of the Cloud Shell pane.
3. In the PowerShell session within the Cloud Shell pane, run the following to create an Azure Key Vault in the resource group **AZ500Lab10-lod57171145**. (If you chose another name for this 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.

powershellTypeCopy

```
$kvName = 'az500kv' + $(Get-Random)
```

```
$location = (Get-AzResourceGroup -ResourceGroupName  
'AZ500Lab10-lod57171145').Location
```

```
New-AzKeyVault -VaultName $kvName -ResourceGroupName 'AZ500Lab10-lod57171145'  
-Location $location -DisableRbacAuthorization
```



- The output of the last command will display the vault name and the vault URI. The vault URI is in the format [https://\(vault_name\).vault.azure.net/](https://(vault_name).vault.azure.net/)
- Close the Cloud Shell pane.
- In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Resource groups** and press the **Enter** key.
- On the **Resource groups** blade, in the list of resource group, click the **AZ500Lab10-1od57171145** (or other name you chose earlier for the resource group) entry.
- On the Resource Group blade, click the entry representing the newly created Key Vault.
- On the Key Vault blade, in the **Overview** section, click **Access Policies** and then click **+ Create**.
- On the **Create an access policy** blade, specify the following settings (leave all others with their default values):

Setting	Value
Configure from template (optional)	Key, Secret, & Certificate Management
Key permissions	click Select all resulting in total of 9 selected permissions
Key permissions/Cryptographic Operations	click Sign resulting in total of 1 selected permissions
Secret permissions	click Select all resulting in total of 7 selected permissions
Certification permissions	click Select all resulting in total of 15 selected permissions
Select principal	On the Principal blade, select your user account, and click Next
Application (optional)	click Next
Review + create	click Create

11. The previous Review + create operation returns to the Access policies page that lists Application, Email, Key Permissions, Secret Permissions, and Certificate Permissions.

The screenshot shows the Azure portal's Resource Groups blade. On the left, there's a sidebar with 'Essentials' and 'Resources' tabs. The 'Resources' tab is selected, showing a list of resources. A blue circle highlights the 'az500kv2120226065' entry under the 'Key vault' category. The right side of the screen displays a 'Key Vault (Implementing Secure Data by setting up Always Encrypted)' blade with instructions and a progress bar.

Instructions:

- will display the vault name and the vault URI. The vault URI is in the format `https://(vault_name).vault.azure.net/`

Resources:

4. Close the Cloud Shell pane.
5. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Resource groups** and press the **Enter** key.
6. On the **Resource groups** blade, in the list of resource group, click the **AZ500Lab10-1od57173201** (or other name you chose earlier for the resource group) entry.
7. On the **Resource Group** blade, click the entry representing the newly created Key Vault.

Progress: 1 Hr 12 Min Remaining

The screenshot shows the 'Access policies' blade for the 'az500kv2120226065' key vault. The left sidebar has 'Access policies' selected. The main area shows a table with one record. A blue circle highlights the 'USER' section in the table. The right side of the screen displays a 'Key Vault (Implementing Secure Data by setting up Always Encrypted)' blade with instructions and a progress bar.

Instructions:

- Certification permissions: total of 15 selected permissions
- Select principal: On the Principal blade, select your user account, and click Next
- Application (optional): click Next
- Review + create: click Create

The previous Review + create operation returns to the Access policies page that lists Application, Email, Key Permissions, Secret Permissions, and Certificate Permissions.

Progress: 1 Hr 9 Min Remaining

Task 2: Add a key to Key Vault

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

-
1. In the Azure portal, open a PowerShell session in the Cloud Shell pane.
 2. Ensure **PowerShell** is selected in the upper-left drop-down menu of the Cloud Shell pane.
 3. In the PowerShell session within the Cloud Shell pane, run the following to add a **software-protected key** to the Key Vault:

```
powershellTypeCopy
```

```
$kv = Get-AzKeyVault -ResourceGroupName 'AZ500Lab10-lod57171145'
```

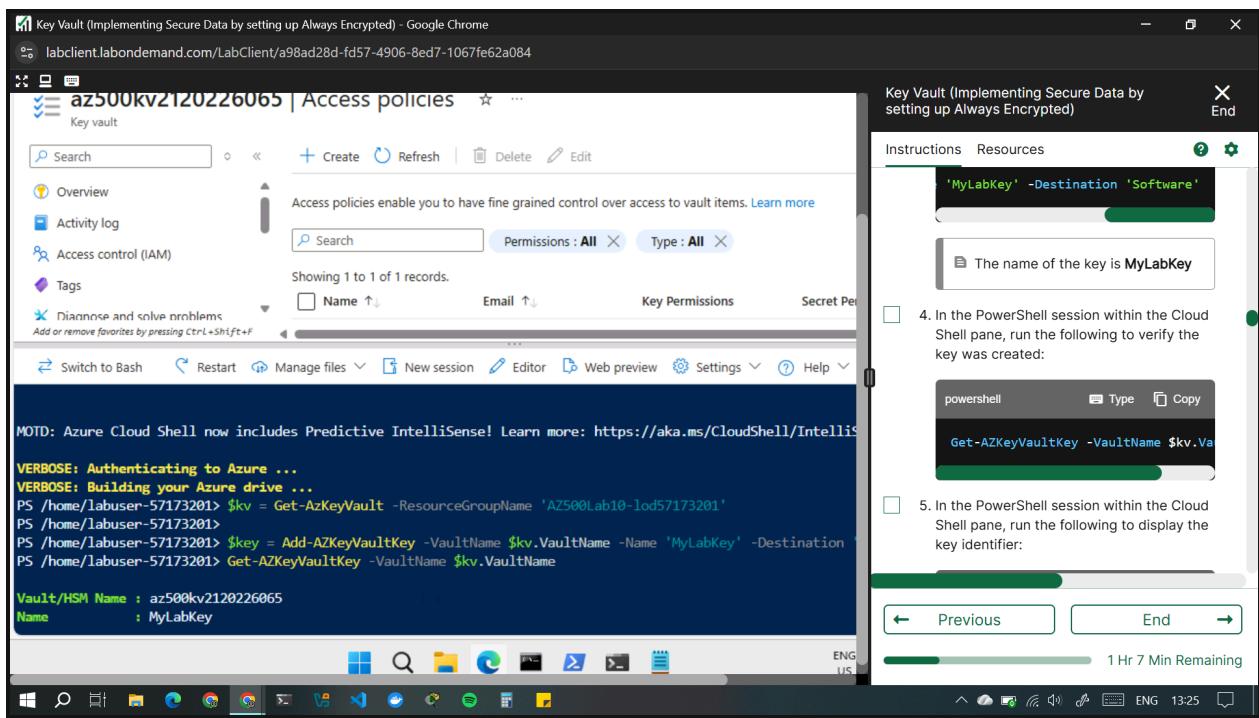
```
$key = Add-AZKeyVaultKey -VaultName $kv.VaultName -Name 'MyLabKey' -Destination  
'Software'
```

*The name of the key is **MyLabKey***

4. In the PowerShell session within the Cloud Shell pane, run the following to verify the key was created:

```
powershellTypeCopy
```

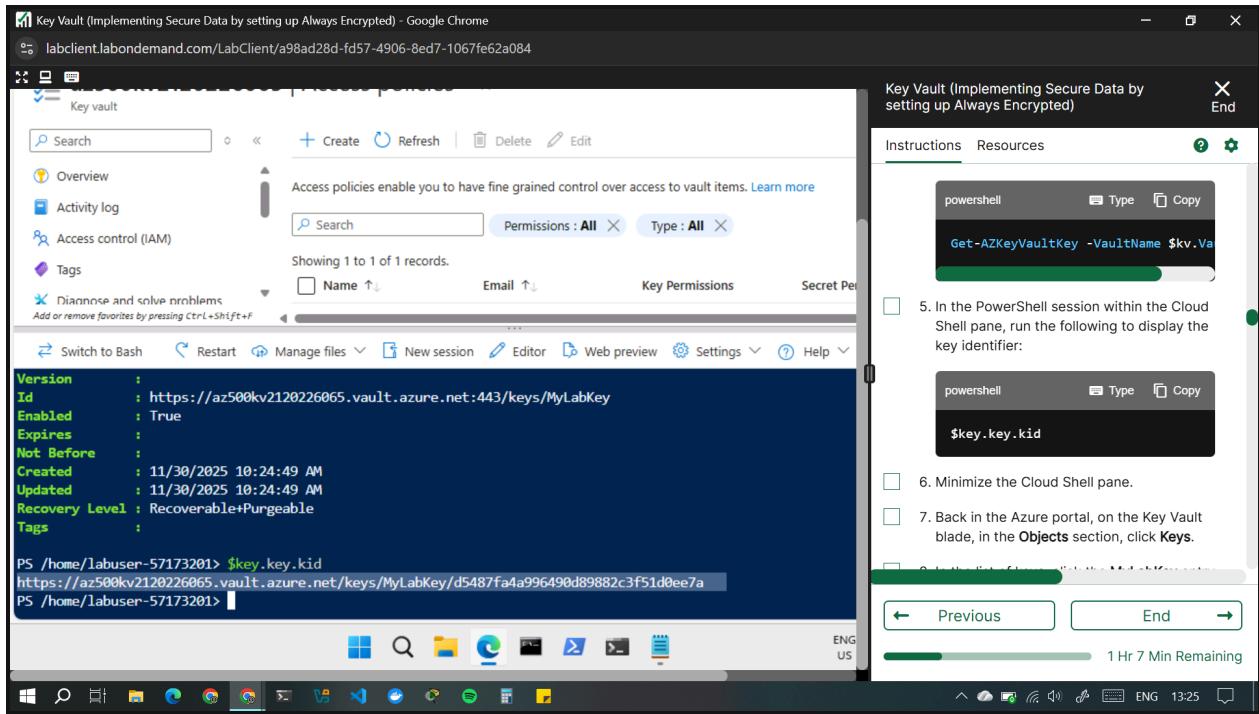
```
Get-AZKeyVaultKey -VaultName $kv.VaultName
```



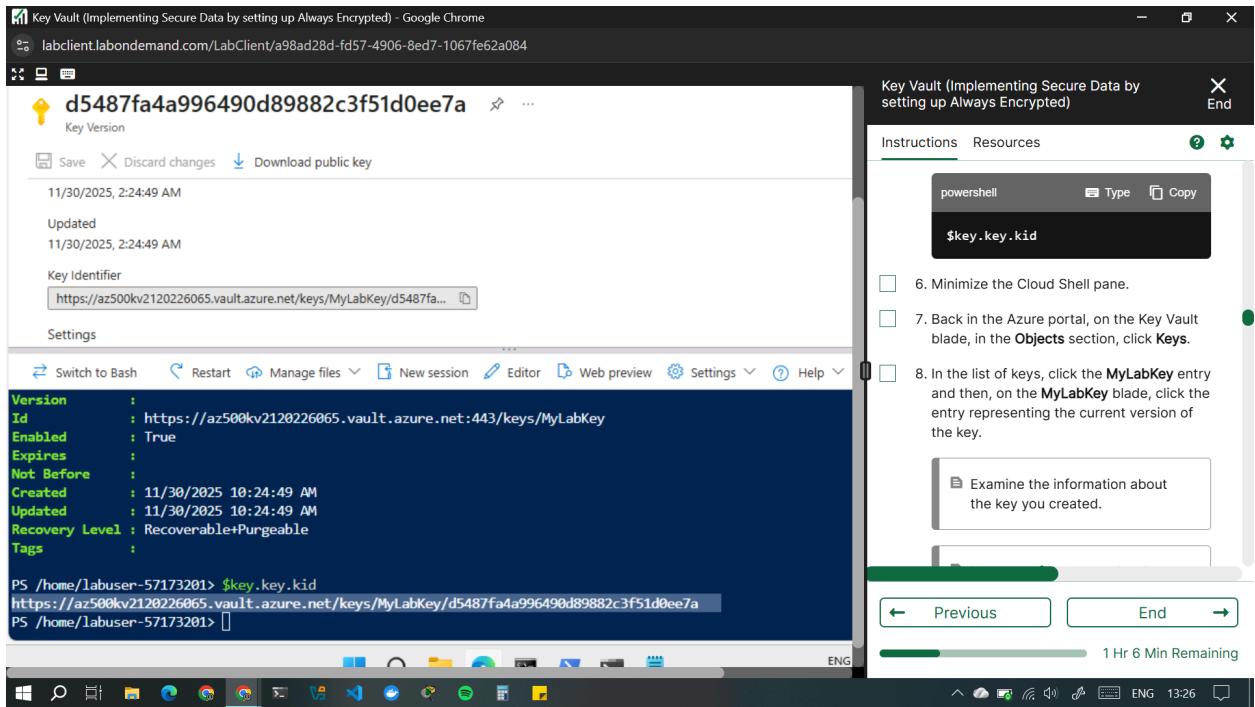
5. In the PowerShell session within the Cloud Shell pane, run the following to display the key identifier:

powershellTypeCopy

\$key.key.kid



You can reference any key by using the key identifier. To get the most current version, reference [https://\(key_vault_name\).vault.azure.net/keys/MyLabKey](https://(key_vault_name).vault.azure.net/keys/MyLabKey) or get the specific version with: [https://.vault.azure.net/keys/MyLabKey/\(key_version\)](https://.vault.azure.net/keys/MyLabKey/(key_version))



- I noted that the Key Identifier in powershell and Azure portal were the same.
- I also noted that the keys created were RSA encryption keys.

Task 3: Add a Secret to Key Vault

Secrets in Azure Key Vault are any sensitive pieces of data, such as passwords, API keys, and connection strings, that are stored securely. Storing secrets in Key Vault removes the need to hardcode them in applications, which enhances security by centralizing credential management and reducing the risk of them being exposed in code.

1. Switch back to the Cloud Shell pane.
2. In the PowerShell session within the Cloud Shell pane, run the following to create a variable with a secure string value:

```
powershellTypeCopy
```

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

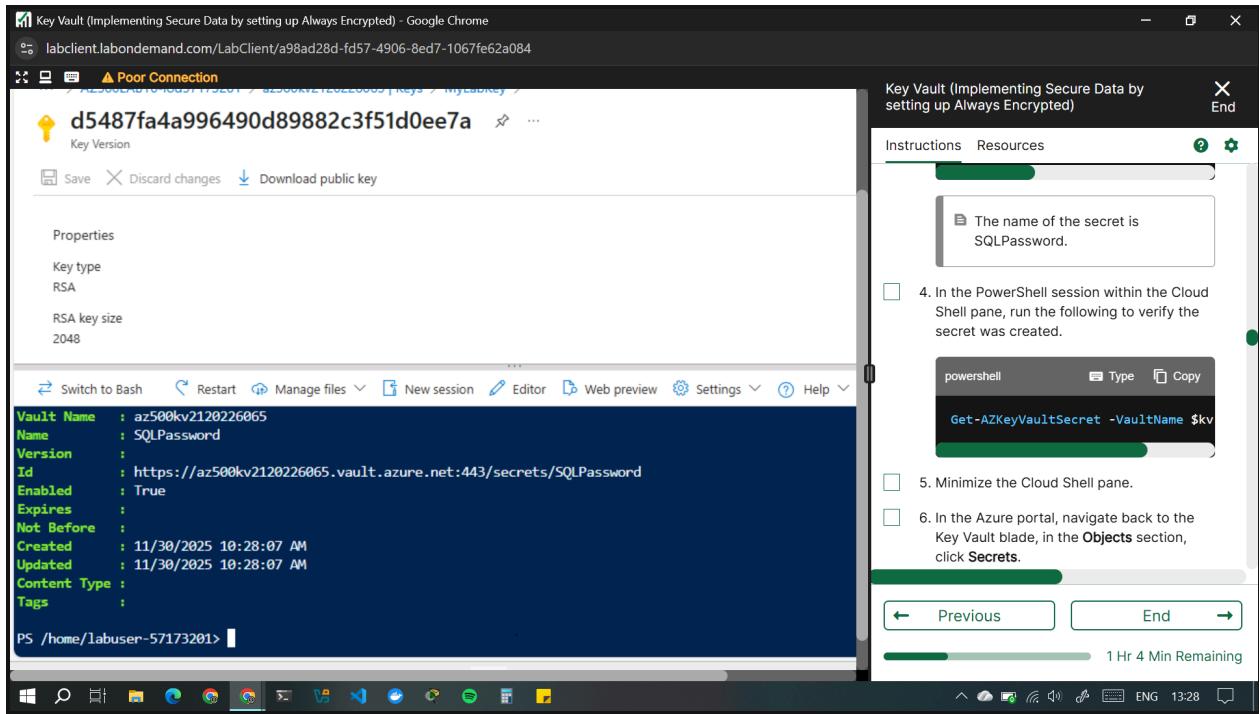
3. In the PowerShell session within the Cloud Shell pane, run the following to add the secret to the vault:

```
powershellTypeCopy  
  
$secret = Set-AZKeyVaultSecret -VaultName $kv.VaultName -Name 'SQLPassword'  
-SecretValue $secretvalue
```

The name of the secret is SQLPassword.

4. In the PowerShell session within the Cloud Shell pane, run the following to verify the secret was created.

```
Get-AZKeyVaultSecret -VaultName $kv.VaultName
```



5. Minimize the Cloud Shell pane.
6. In the Azure portal, navigate back to the Key Vault blade, in the **Objects** section, click **Secrets**.
7. In the list of secrets, click the **SQLPassword** entry and then, on the **SQLPassword** blade, click the entry representing the current version of the secret.

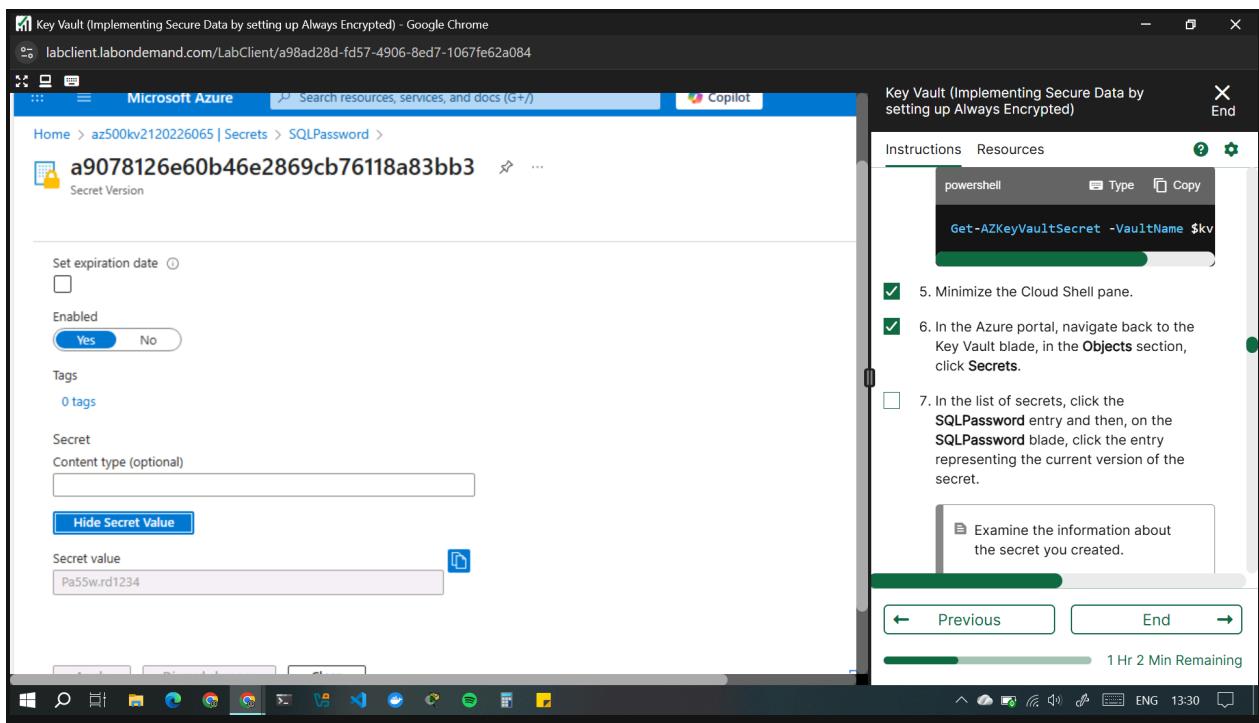
Examine the information about the secret you created.

To get the most current version of a secret, reference

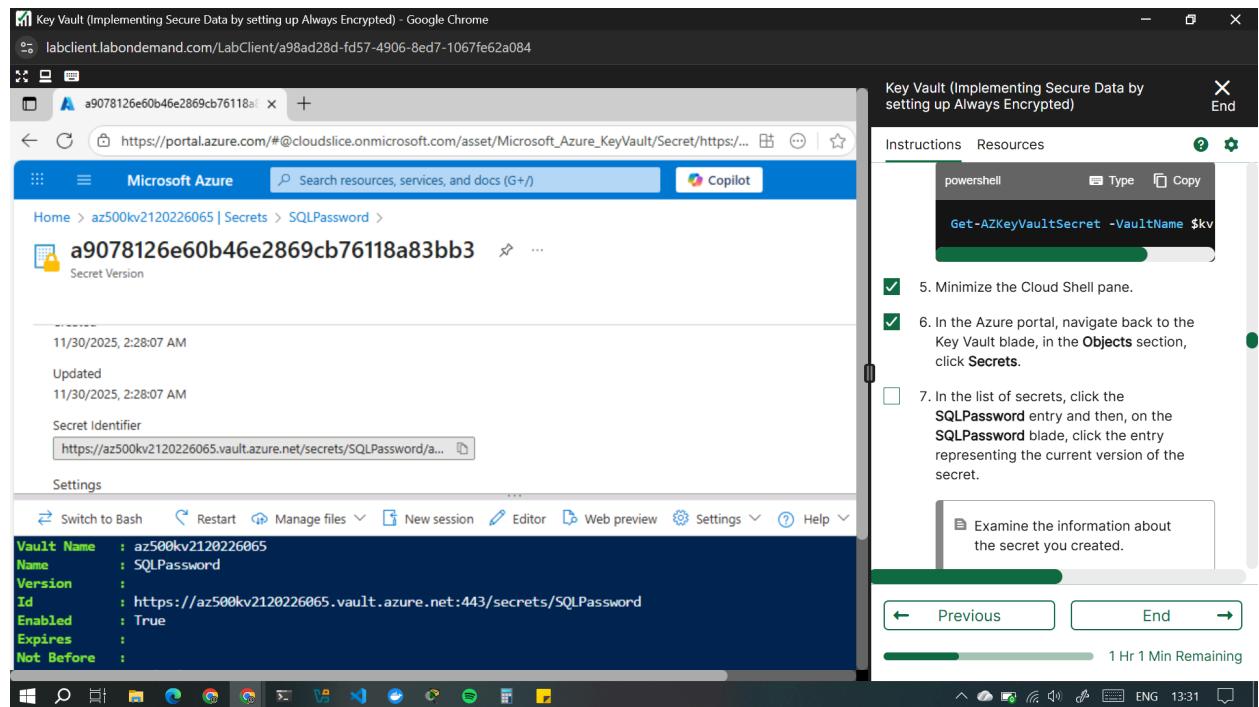
https://<key_vault_name>.vault.azure.net/secrets/<secret_name> or get a

specific version, reference

https://<key_vault_name>.vault.azure.net/secrets/<secret_name>/<secret_version>



- The secret 'SQLPassword' can be seen with its secret value 'Pa55w.rd1234' in Azure Key Vault



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

In this exercise, I completed the following tasks:

- Task 1: Enable a client application to access the Azure SQL Database service.
- Task 2: Create a policy allowing the application access to the Key Vault.
- Task 3: Retrieve SQL Azure database ADO.NET Connection String
- Task 4: Log on to the Azure VM running Visual Studio 2019 and SQL Management Studio 19
- Task 5: Create a table in the SQL Database and select data columns for encryption

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

In this task, you will enable a client application to access the Azure SQL Database service. This will be done by setting up the required authentication and acquiring the Application ID and Secret that you will need to authenticate your application.

1. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **App Registrations** and press the **Enter** key.
2. On the **App Registrations** blade, click **+ New registration**.
3. On the **Register an application** blade, specify the following settings (leave all others with their default values):

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.

The screenshot shows a Microsoft Edge browser window with the URL labclient.labondemand.com/LabClient/a98ad28d-fd57-4906-8ed7-1067fe62a084. The page displays the 'sqlApp' blade under the 'Overview' section. Key details shown include:

- Display name:** sqlApp
- Application (client) ID:** 44283766-c2bd-4d6a-88a7-5e66bd66e5b4 (highlighted with a tooltip 'Copied')
- Object ID:** a56c0781-5e59-4066-9696-a93672fcf4d5
- Directory (tenant) ID:** 8eb87a6e-8055-4135-b69d-f19c799ec045
- Supported account types:** My organization only

A floating instruction card provides steps for the task:

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, click Certificates & secrets.
7. On the sqlApp | Certificates & secrets blade / Client Secrets section, click + New client secret

6. On the **sqlApp** blade, in the **Manage** section, click **Certificates & secrets**.

7. On the **sqlApp | Certificates & secrets** blade / **Client Secrets** section, click **+ New client secret**

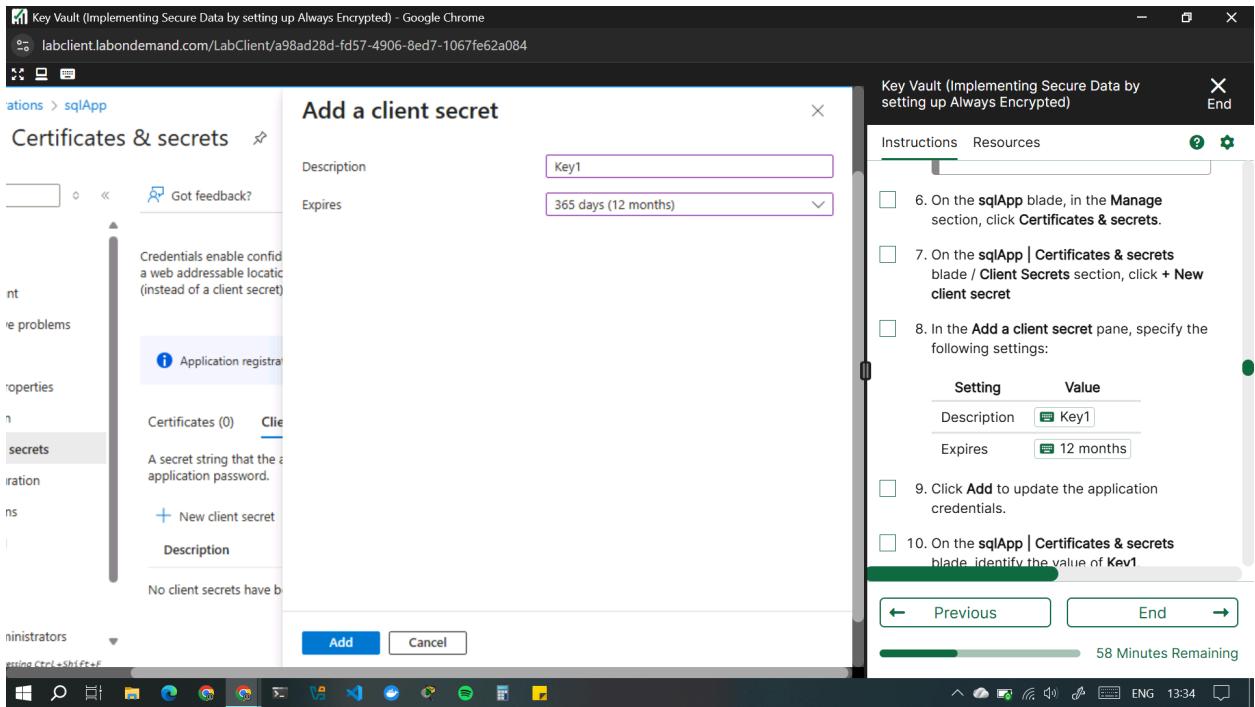
8. In the **Add a client secret** pane, specify the following settings:

Setting	Value
---------	-------

Description Key1

n

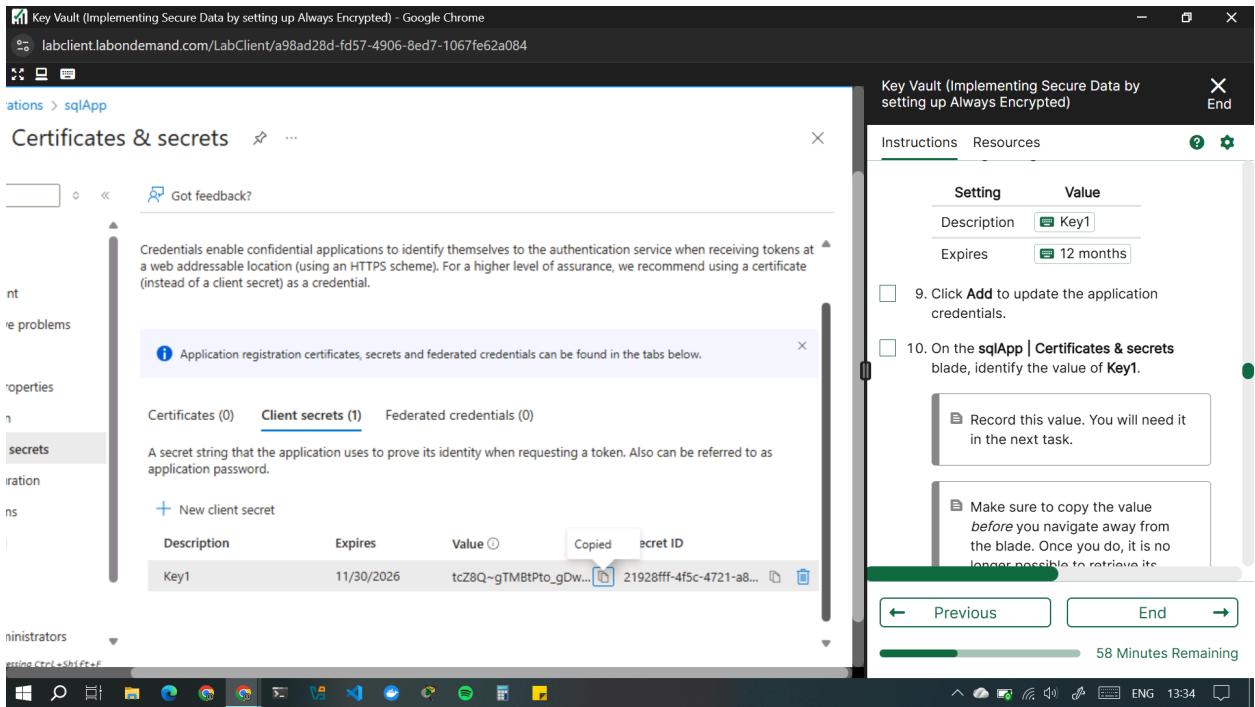
Expires 12 months



9. Click **Add** to update the application credentials.
10. On the **sqlApp | Certificates & secrets** blade, identify the value of **Key1**.

Record this value. You will need it in the next task.

Make sure to copy the value *before* you navigate away from the blade. Once you do, it is no longer possible to retrieve its clear text value.

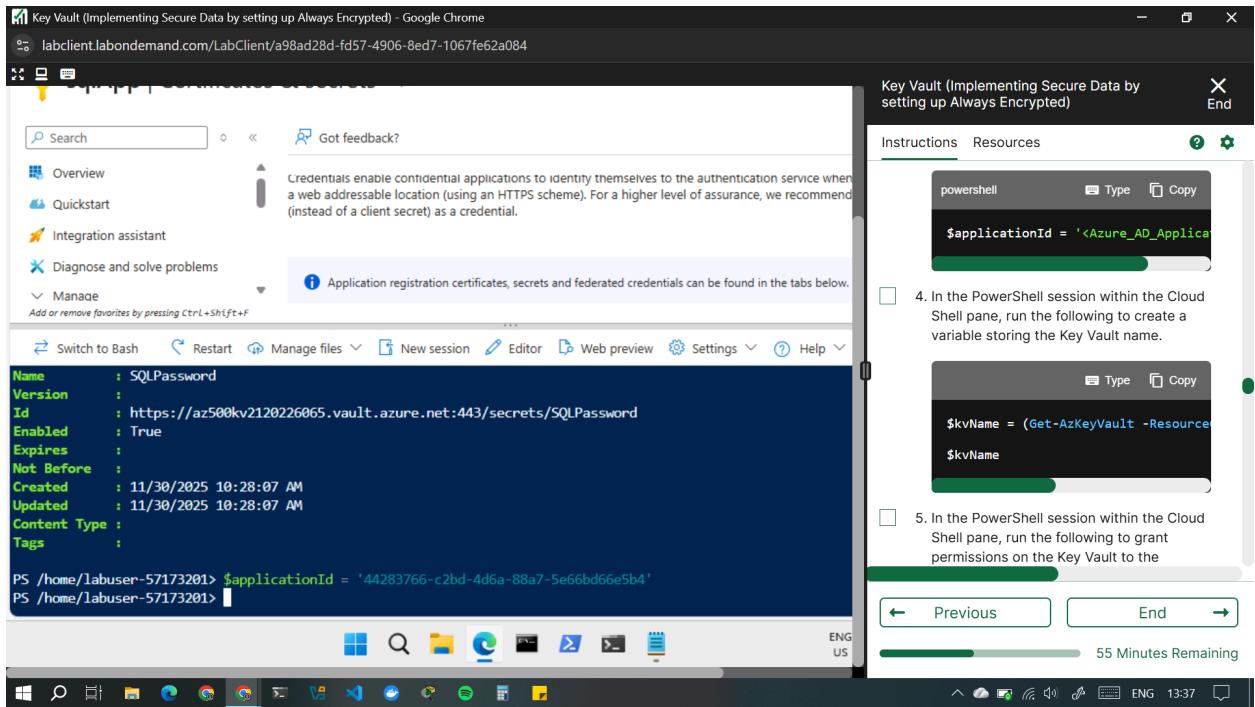


Task 2: Create a policy allowing the application access to the Key Vault.

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

1. In the Azure portal, open a PowerShell session in the Cloud Shell pane.
2. Ensure **PowerShell** is selected in the upper-left drop-down menu of the Cloud Shell pane.
3. In the PowerShell session within the Cloud Shell pane, run the following to create a variable storing the **Application (client) ID** you recorded in the previous task (replace the (Azure_AD_Application_ID) placeholder with the value of the **Application (client) ID**):

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



4. In the PowerShell session within the Cloud Shell pane, run the following to create a variable storing the Key Vault name.

TypeCopy

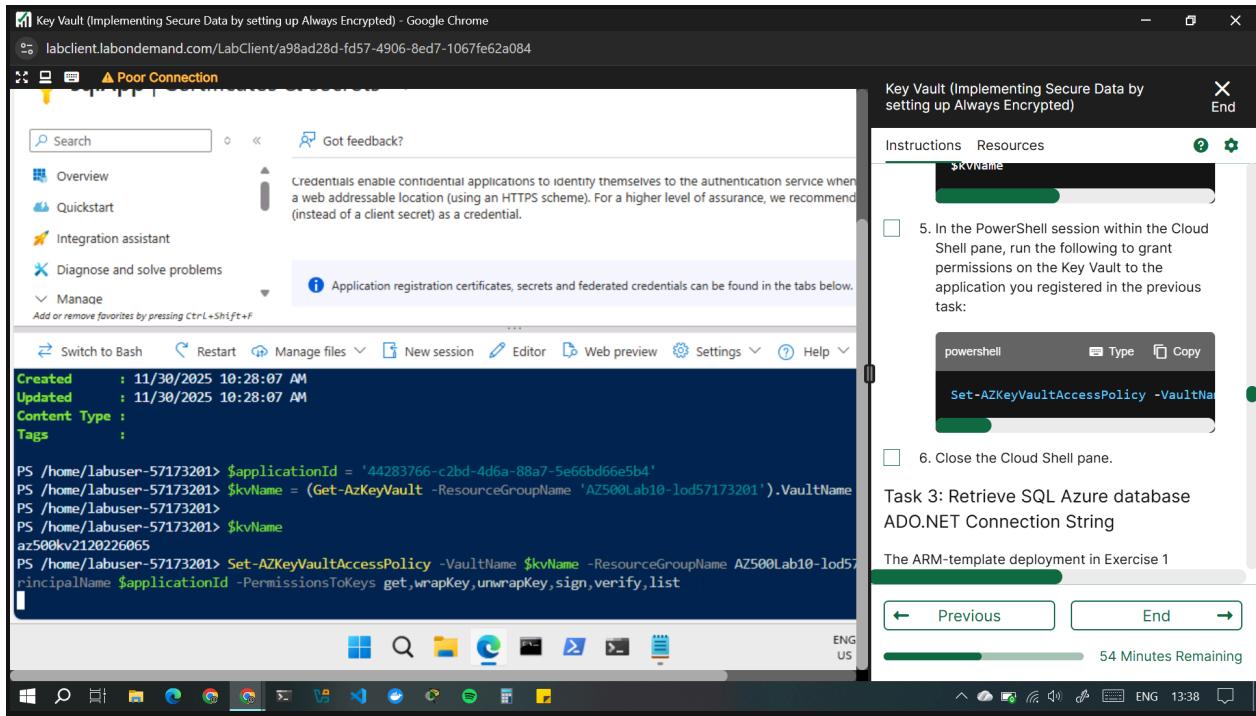
```
$kvName = (Get-AzKeyVault -ResourceGroupName 'AZ500Lab10-lod57171145').VaultName
```

\$kvName

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:

powershellTypeCopy

```
Set-AZKeyVaultAccessPolicy -VaultName $kvName -ResourceGroupName
AZ500Lab10-lod57171145 -ServicePrincipalName $applicationId
-PermissionsToKeys get,wrapKey,unwrapKey,sign,verify,list
```



6. Close the Cloud Shell pane.

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

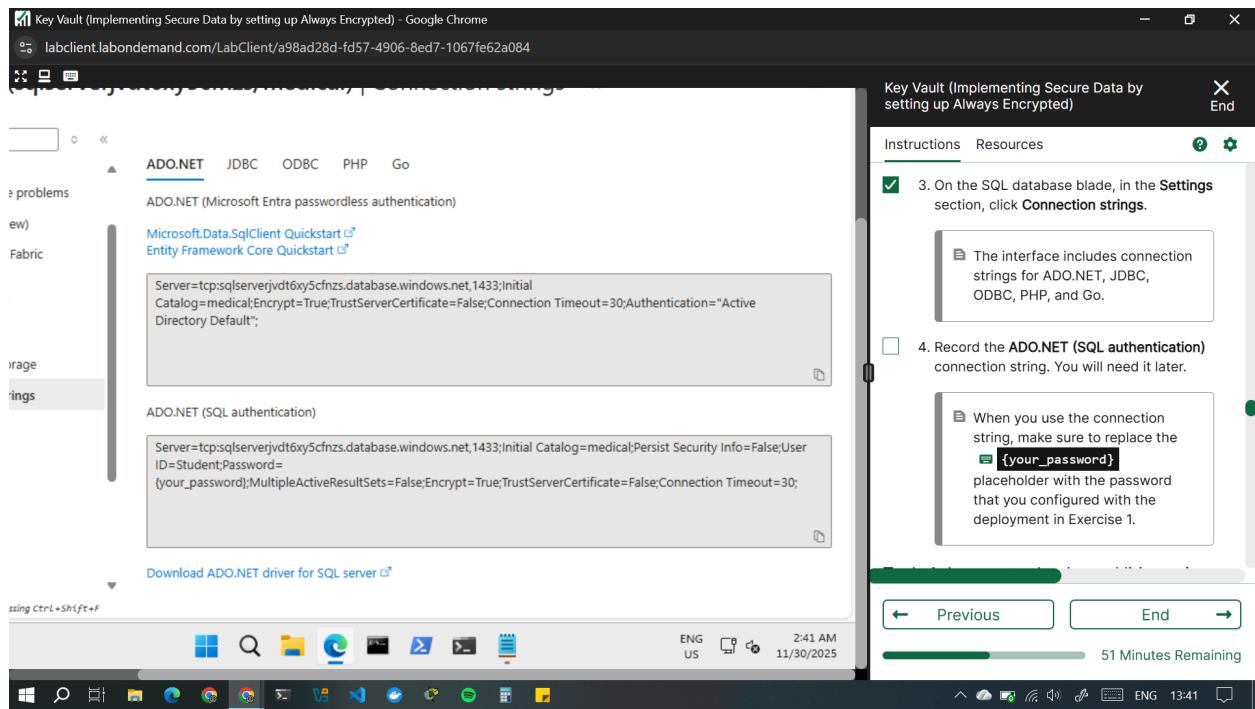
The ARM-template deployment in Exercise 1 provisioned an Azure SQL Server instance and an Azure SQL database named **medical**. You will update the empty database resource with a new table structure and select data columns for encryption

1. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **SQL databases** and press the **Enter** key.
2. In the list of SQL databases, click the **medical()** entry.
3. On the SQL database blade, in the **Settings** section, click **Connection strings**.

The interface includes connection strings 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.

1. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **virtual machines** and press the **Enter** key.

2. In the list of Virtual Machines shown, select the **az500-10-vm1** entry. On the **az500-10-vm1** blade, on the **Essentials** pane, take note of the **Public IP address**. You will use this later.

The screenshot shows the Azure portal interface for managing a virtual machine named 'az500-10-vm1'. The left sidebar contains navigation links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Connect, Networking, Settings, Availability + scale, Security, Backup + disaster recovery, Operations, and Monitoring. The main content area displays the 'az500-10-vm1' blade with the 'Essentials' section expanded. Key details include:

- Resource group (move):** AZ500Lab10-lod57173201
- Operating system:** Windows (Windows Server 2022 Datacenter)
- Status:** Running
- Location:** East US
- Subscription (move):** AZ-500T00-A CSR_1
- Subscription ID:** 25761271-1ab0-4f3c-8ef1-be9b5ddede1d
- Primary NIC public IP:** 52.186.48.22
- Associated public IPs:** 1
- Virtual network/subnet:** az500-10-vnet1/subnet0
- DNS name:** Not configured
- Health state:** -
- Time created:** 11/30/2025, 10:12 AM UTC

A callout box on the right provides instructions for setting up Always Encrypted in the Azure Key Vault:

- In the Azure portal, in the Search resources, services, and docs text box at the top of the Azure portal page, type **virtual machines** and press the **Enter** key.
- In the list of Virtual Machines shown, select the **az500-10-vm1** entry. On the **az500-10-vm1** blade, on the **Essentials** pane, take note of the **Public IP address**. You will use this later.

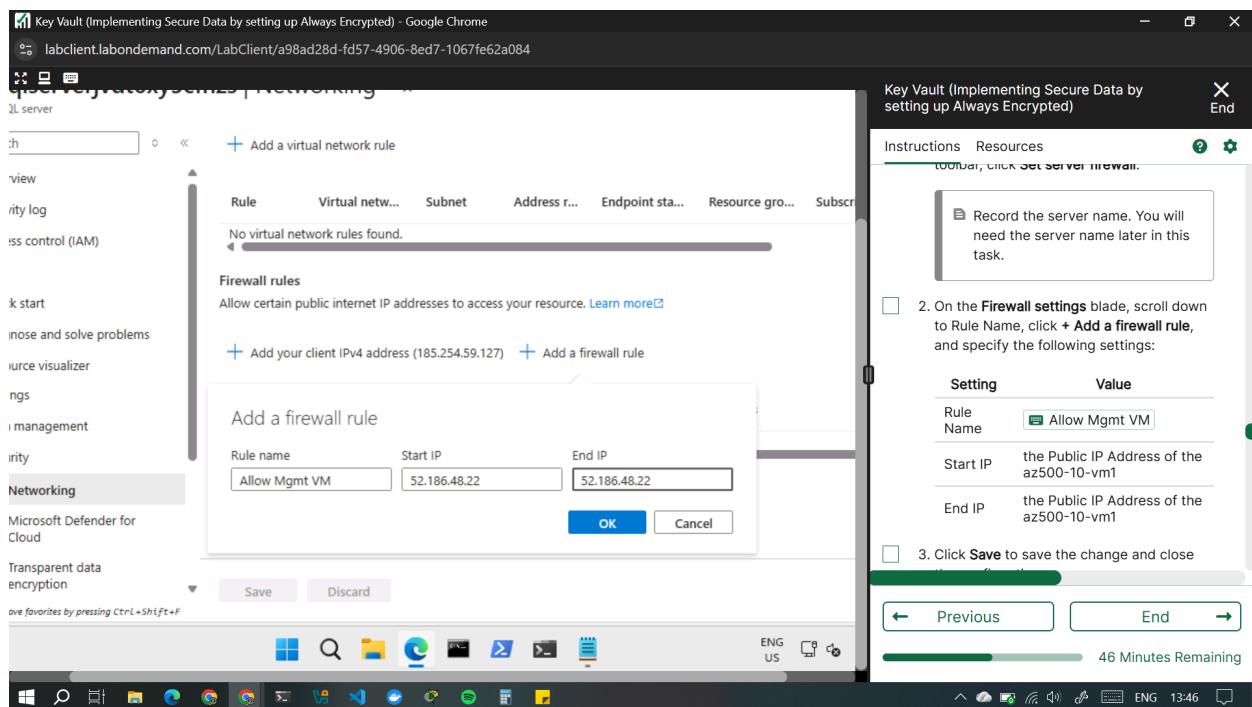
Task 5: Create a table in the SQL Database and select data columns for encryption

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

In this task, you will connect to the SQL Database with SQL Server Management Studio and create a table. You will then encrypt two data columns using an autogenerated key from the Azure Key Vault.

- In the Azure portal, navigate to the blade of the **medical** SQL database, in the **Essentials** section, identify the **Server name** (copy to clipboard), and then, in the toolbar, click **Set server firewall**. Record the server name. You will need the server name later in this task.
- On the **Firewall settings** blade, scroll down to Rule Name, click **+ Add a firewall rule**, and specify the following settings:

Setting	Value
Rule	Allow Mgmt VM
Name	
Start IP	the Public IP Address of the az500-10-vm1
End IP	the Public IP Address of the az500-10-vm1



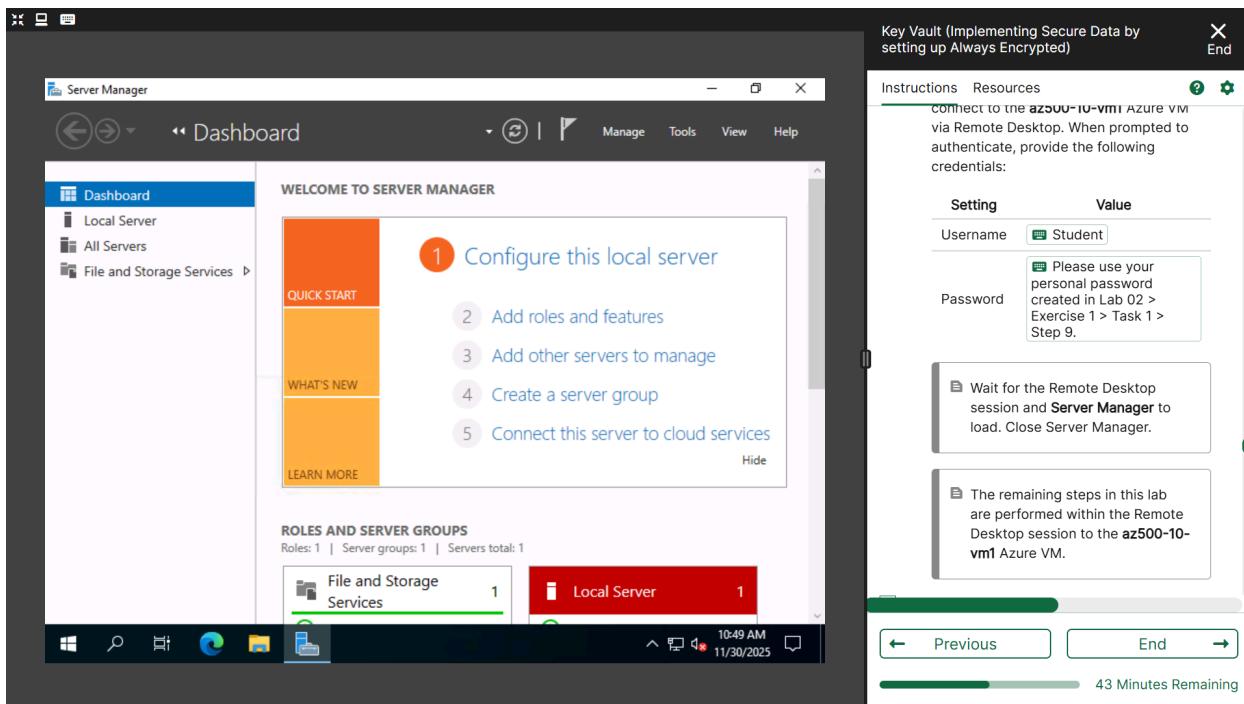
- Click **Save** to save the change and close the confirmation pane.
- This modifies the server firewall settings, allowing connections to the medical database from the Azure VM's public IP address you deployed in this lab.
- Navigate back to the **az500-10-vm1** blade, click **Overview**, next click **Connect** and, in the drop down menu, click **Connect**.

5. Download the 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	Student
Password	Please use your personal password created in Lab 02 > Exercise 1 > Task 1 > Step 9.

6. Wait for the Remote Desktop session and **Server Manager** to load. Close Server Manager.

The remaining steps in this lab are performed within the Remote Desktop session to the **az500-10-vm1** Azure VM.



7. Install

<https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?preserve-view=true&view=sql-server-2017> on **az500-10-vm1**. Azure VM.

8. Open **SQL Server Management Studio**.

9. In the **Connect to Server** dialog box, specify the following settings:

Setting	Value
Server Type	Database Engine
Server Name	the server name you identified earlier in this task
Authentication	SQL Server Authentication
Username	Student
Password	Please use your personal password created in Lab 02 > Exercise 2 > Task 1 > Step 3.

10. In the **Connect to Server** dialog box, click **Connect**.

11. Within the **SQL Server Management Studio** console, in the **Object Explorer** pane, expand the **Databases** folder.

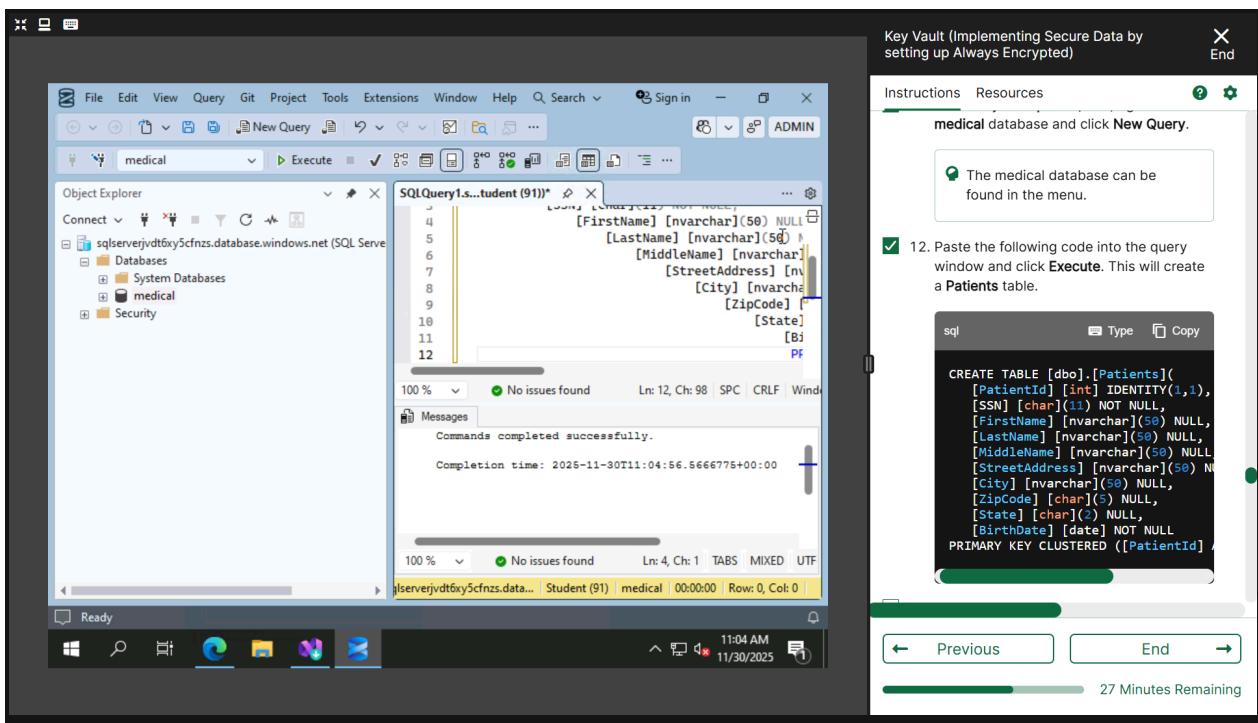
12. In the **Object Explorer** pane, right-click the **medical** database and click **New Query**.

The medical database can be found in the menu.

13. Paste the following code into the query window and click **Execute**. This will create a **Patients** table.

```
sqlTypeCopy
```

```
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] );
```



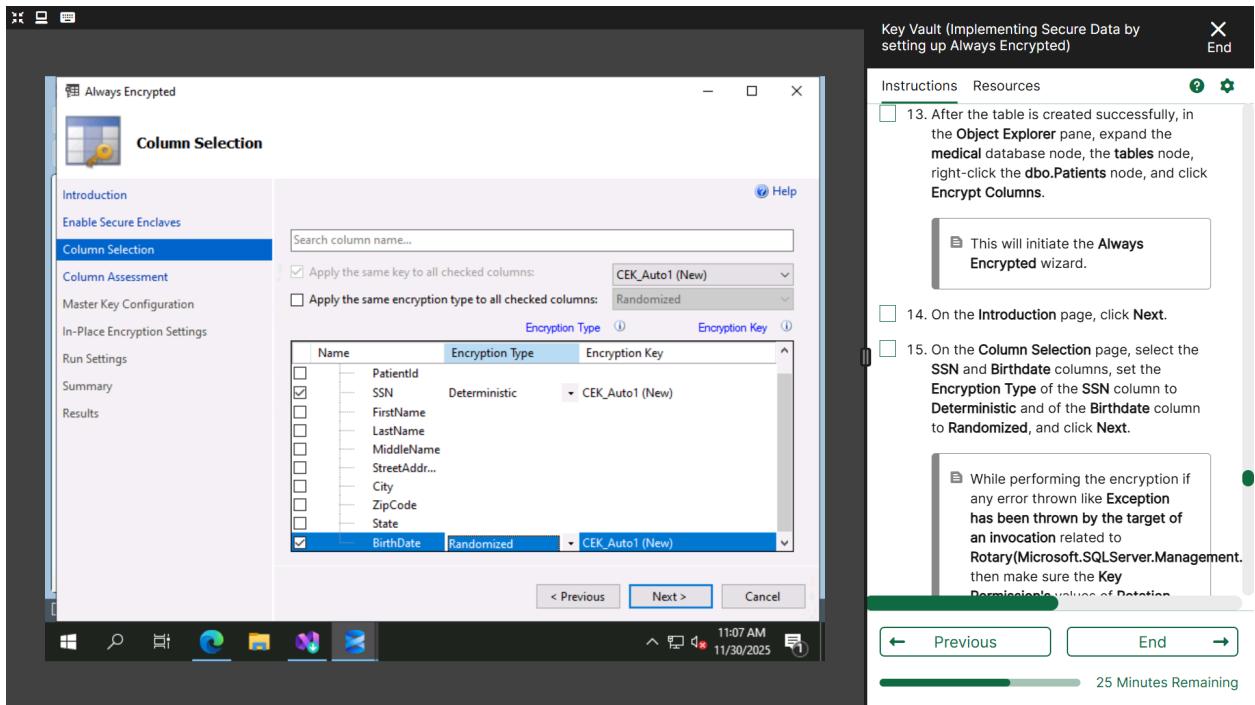
14. After the table is created successfully, in the **Object Explorer** pane, expand the **medical** database node, the **tables** node, right-click the **dbo.Patients** node, and click **Encrypt Columns**.

This will initiate the **Always Encrypted** wizard.

14. On the **Introduction** page, click **Next**.

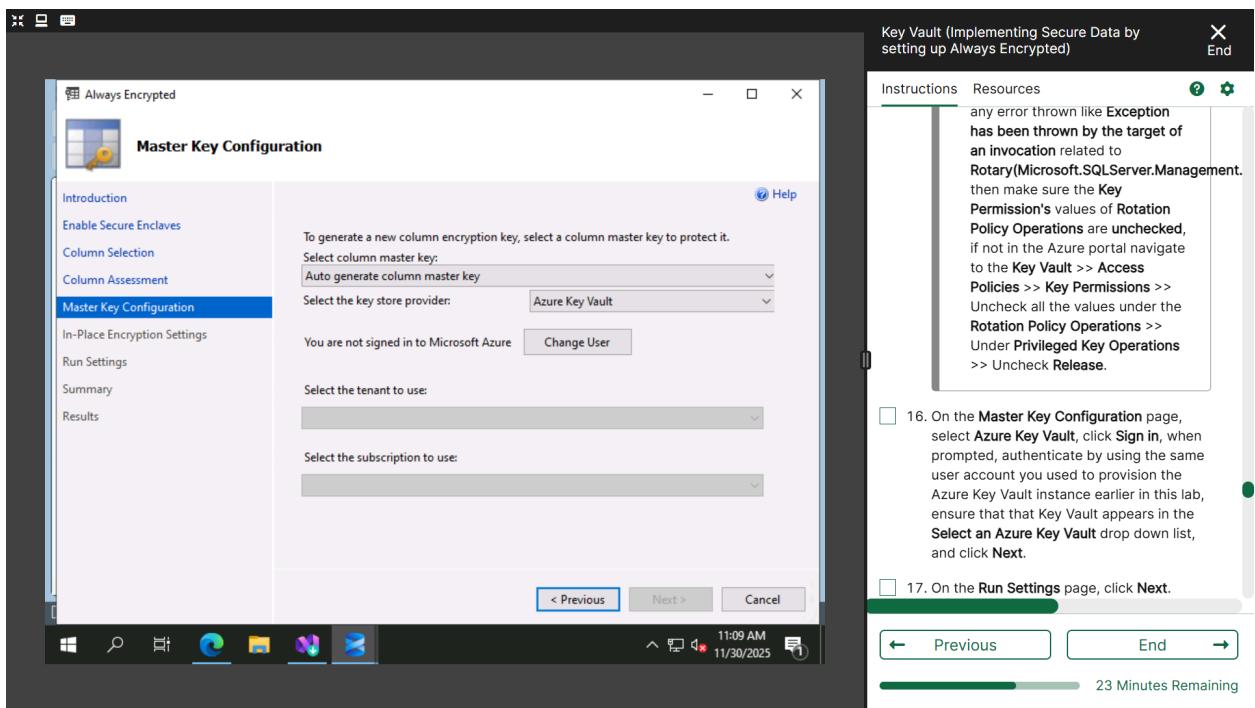
15. On the **Column Selection** page, select the **SSN** and **Birthdate** columns, set the **Encryption Type** of the **SSN** column to **Deterministic** and of the **Birthdate** column to **Randomized**, and click **Next**.

- **Deterministic Encryption:** Always produces the same encrypted value for a given plaintext value. This means if you encrypt "ABC" twice, the resulting ciphertext will be identical each time.
- **Randomized Encryption:** Generates a different encrypted value for the same plaintext value each time it's encrypted. This is achieved by using a randomly generated Initialization Vector (IV) for each encryption operation.

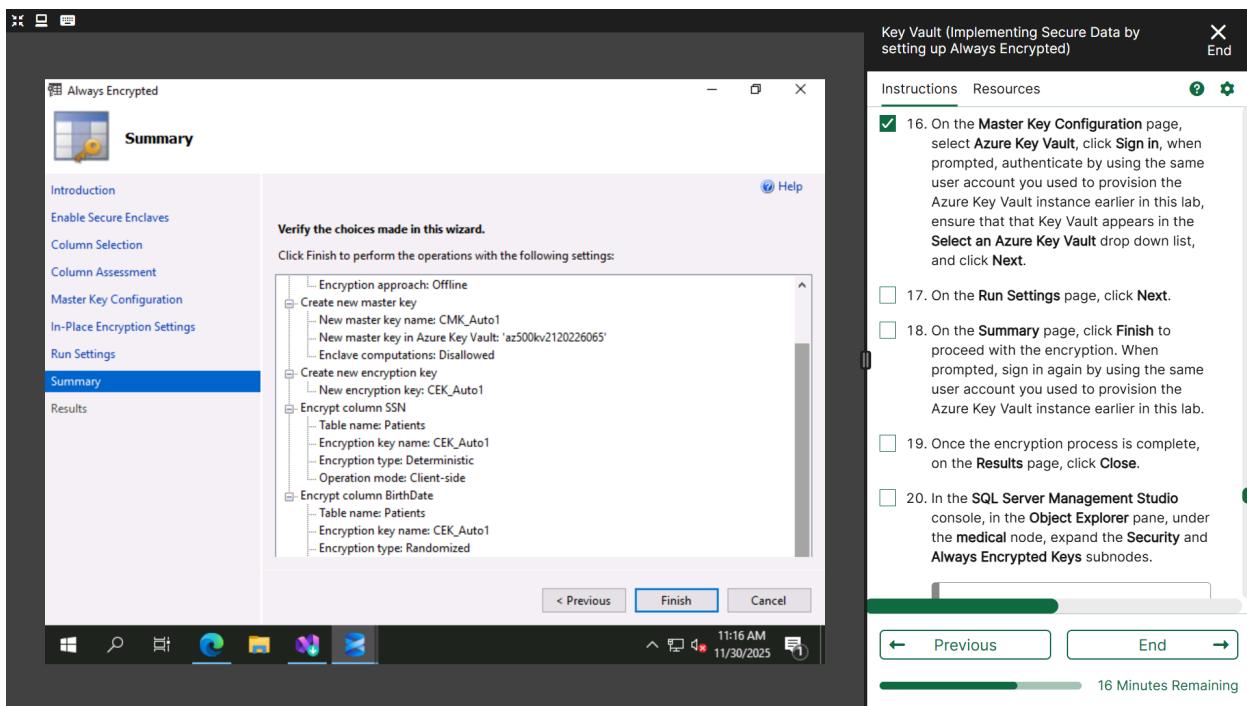


*While performing the encryption if any error thrown like **Exception has been thrown by the target of an invocation** related to **Rotary(Microsoft.SqlServer.Management.ServiceManagement)** then make sure the **Key Permission's values of Rotation Policy Operations** are **unchecked**, if not in the 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, ensure that that Key Vault appears in the **Select an Azure Key Vault** drop down list, and click **Next**.



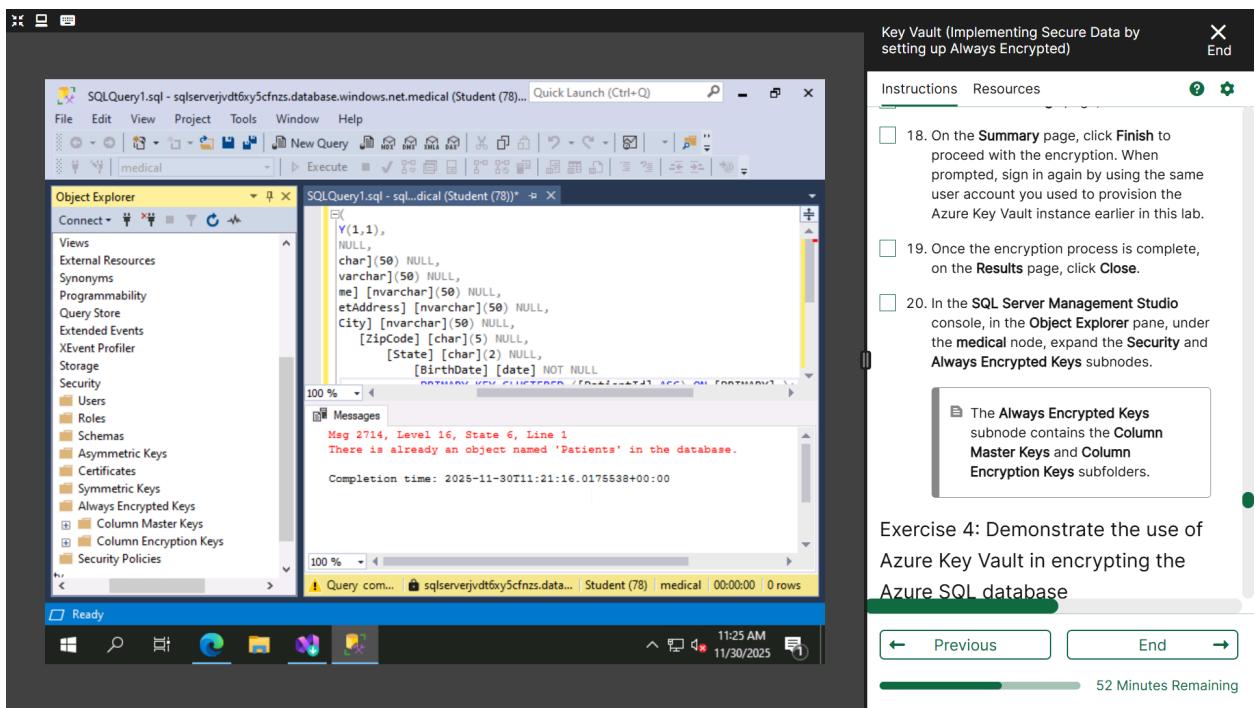
17. On the **Run Settings** page, click **Next**.
18. On the **Summary** page, click **Finish** to proceed with the encryption. When prompted, sign in again by using the same user account you used to provision the Azure Key Vault instance earlier in this lab.



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.



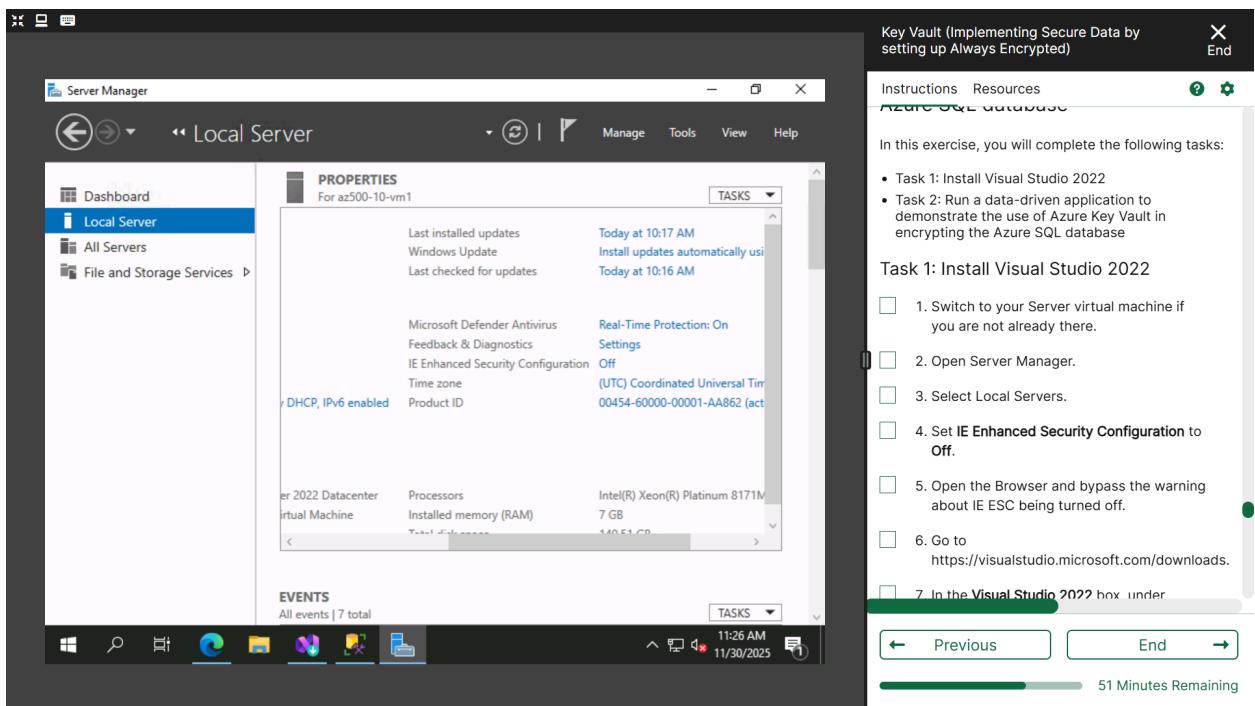
Exercise 4: Demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

In this exercise, I completed the following tasks:

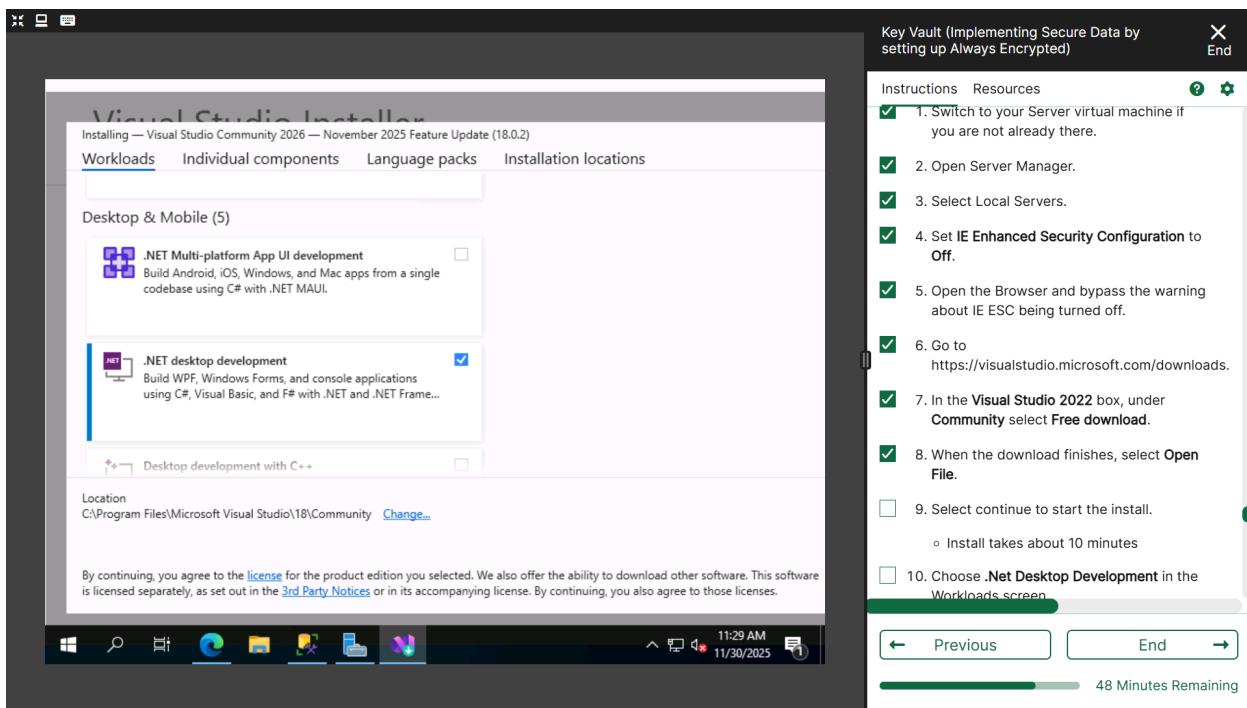
- Task 1: Install Visual Studio 2022
- Task 2: Run a data-driven application to demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

Task 1: Install Visual Studio 2022

1. Switch to your Server virtual machine if you are not already there.
2. Open Server Manager.
3. Select Local Servers.
4. Set **IE Enhanced Security Configuration** to **Off**.



5. Open the Browser and bypass the warning about IE ESC being turned off.
 6. Go to <https://visualstudio.microsoft.com/downloads>.
 7. In the **Visual Studio 2022** box, under **Community** select **Free download**.
 8. When the download finishes, select **Open File**.
 9. Select continue to start the install.
 - Install takes about 10 minutes
10. Choose **.Net Desktop Development** in the Workloads screen.



Task 2: Run a data-driven application to demonstrate the use of Azure Key Vault in encrypting the Azure SQL database

You 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**.

1. From the RDP session to the **az500-10-vm1**, launch **Visual Studio 2019** from the **Start menu**.
2. Switch to the window displaying Visual Studio 2019 welcome message, click the **Sign in** button and, when prompted, provide the credentials you used to authenticate to the Azure subscription you are using in this lab.
3. On the **Get started** page, click **Create a new project**.
4. In the list of project templates, search for **Console App (.NET Framework)**, in the list of results, click **Console App (.NET Framework)** for **C#**, and click **Next**.

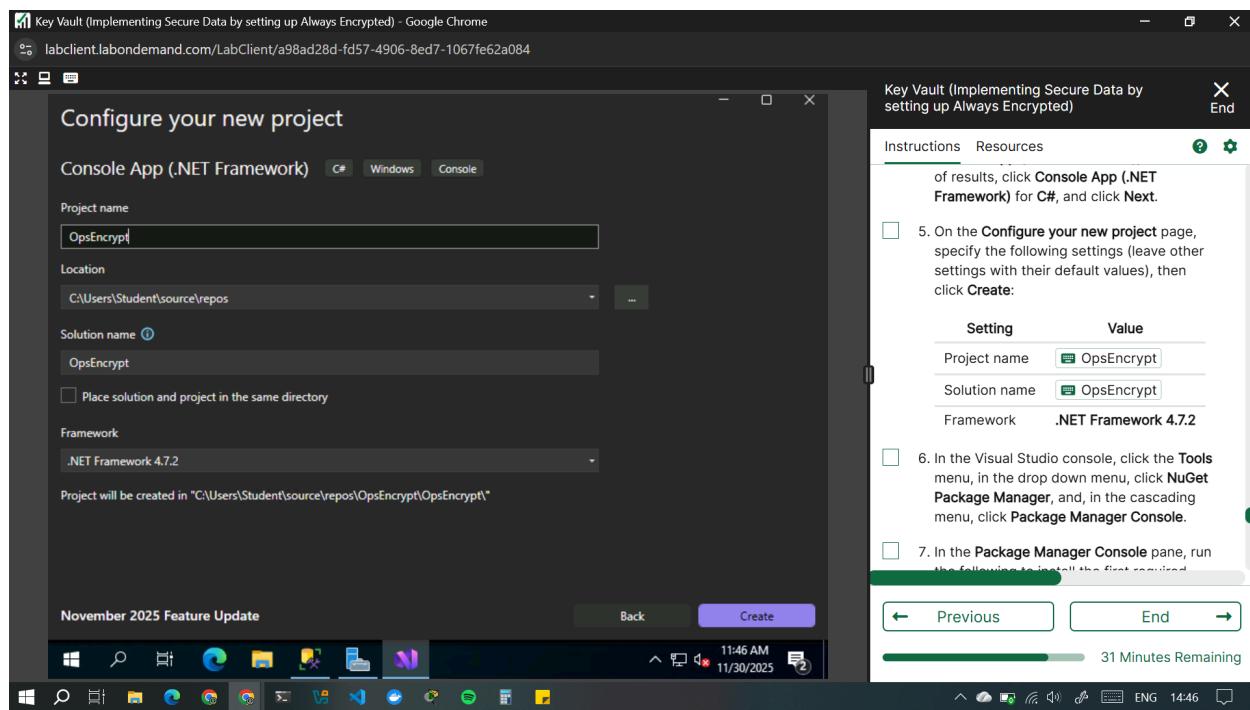
5. On the **Configure your new project** page, specify the following settings (leave other settings with their default values), then click **Create**:

Setting	Value
---------	-------

Project name OpsEncrypt

Solution name OpsEncrypt

Framework **.NET Framework 4.7.2**

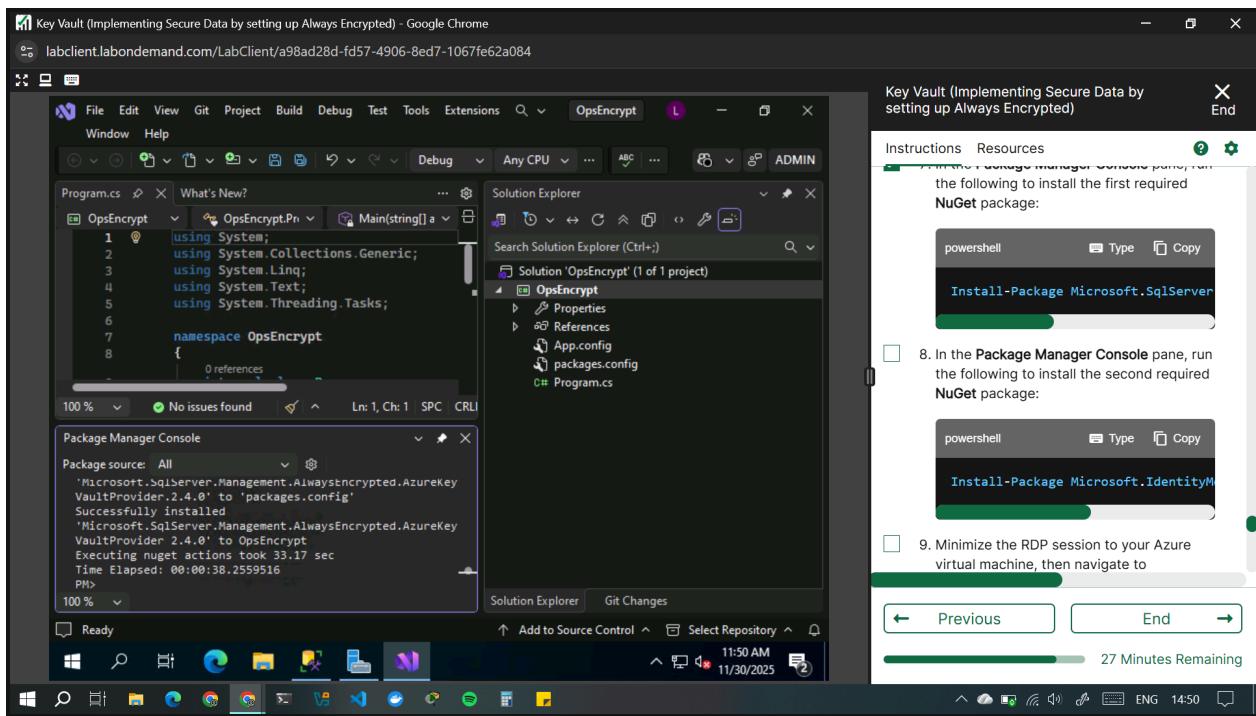


6. In the Visual Studio console, click the **Tools** menu, in the drop down menu, click **NuGet Package Manager**, and, in the cascading menu, click **Package Manager Console**.

- NuGet is the official package manager for .NET, enabling developers to create, share, and consume reusable .NET libraries and components. These libraries are distributed as "packages," which are essentially single ZIP files with a .nupkg extension.

7. In the **Package Manager Console** pane, run the following to install the first required **NuGet** package:

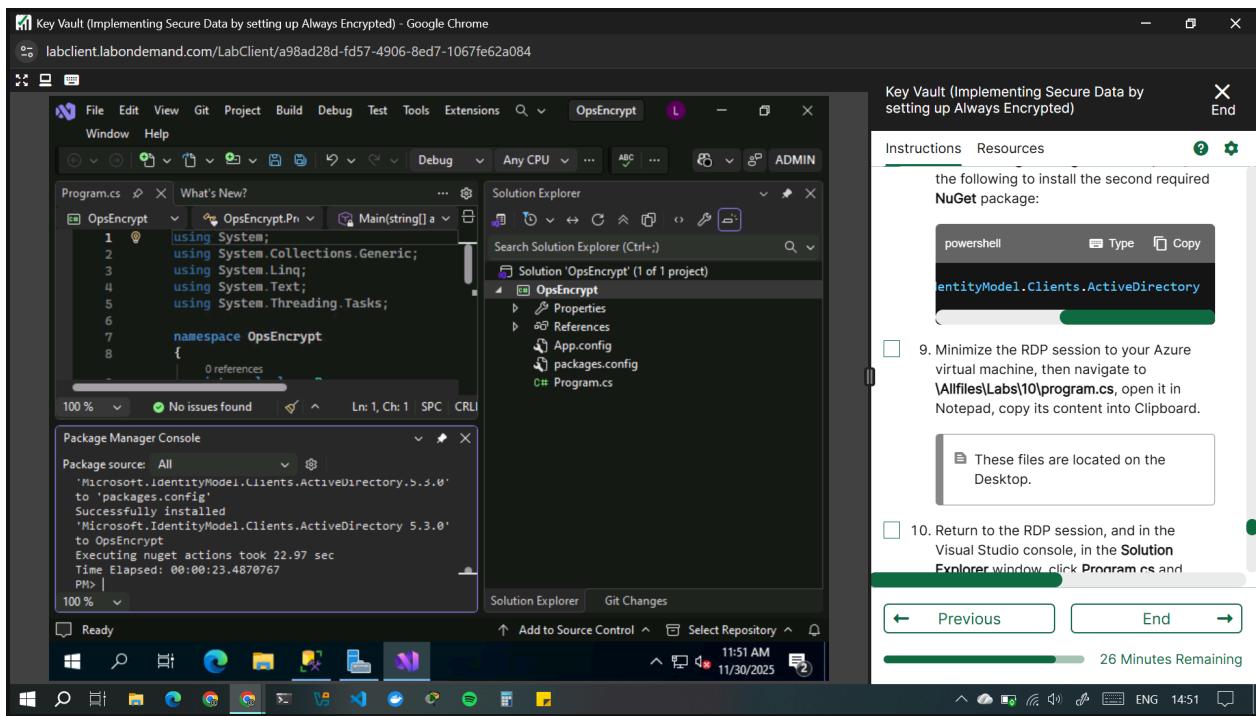
```
powershellTypeCopy  
  
Install-Package  
Microsoft.SqlServer.Management.AlwaysEncrypted.AzureKeyVaultProvider
```



8. In the **Package Manager Console** pane, run the following to install the second required **NuGet** package:

powershellTypeCopy

Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory

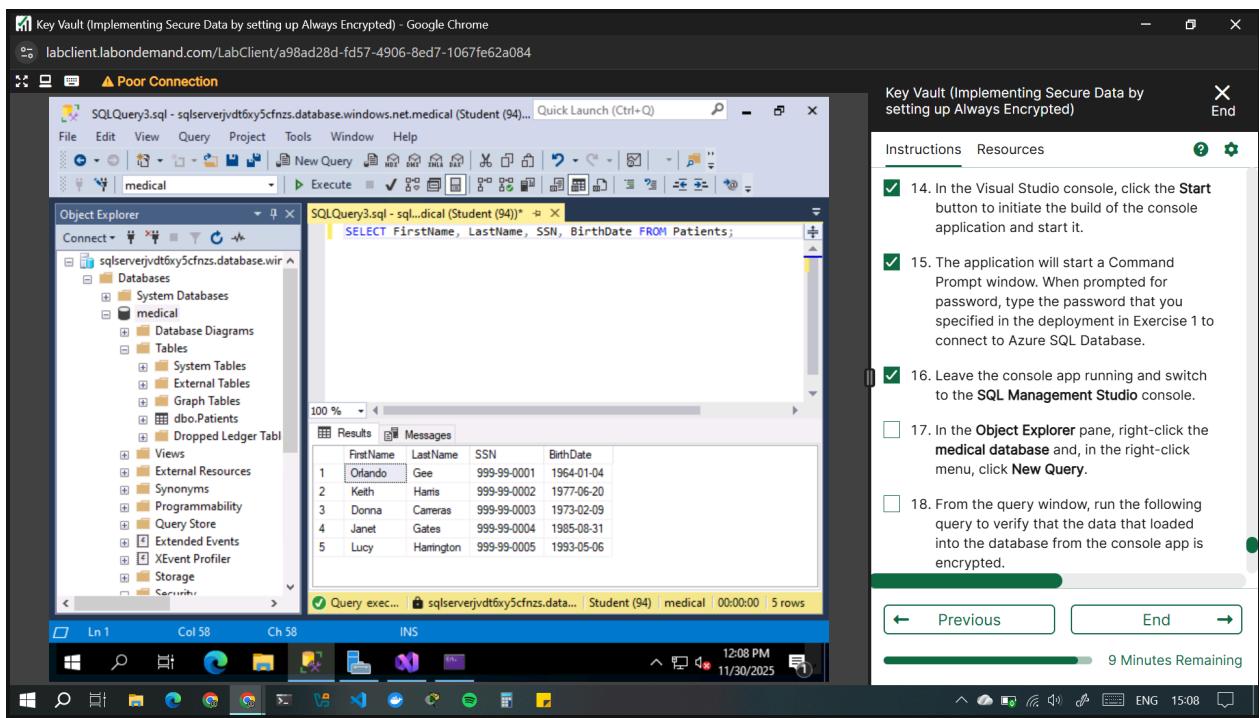


9. Minimize the RDP session to your Azure virtual machine, then navigate to **\Allfiles\Labs\10\program.cs**, open it in Notepad, copy its content into Clipboard.
These files are located on the Desktop.
10. Return to the RDP session, and in the Visual Studio console, in the **Solution Explorer** window, click **Program.cs** and replace its content with the code you copied into Clipboard.
11. In the Visual Studio window, in the **Program.cs** pane, in line 15, replace the (connection string noted earlier) placeholder with the Azure SQL database **ADO.NET** connection string you recorded earlier in the lab. In the connection string, replace the **{your_password}** placeholder, with the password that you specified in the deployment in Exercise 1. If you saved

-
- the string on the lab computer, you may need to leave the RDP session to copy the ADO string, then return to the Azure virtual machine to paste it in.
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, replace the (key value noted earlier) placeholder with the the value of **Key1** of the registered app you recorded earlier in the lab.
 14. In the Visual Studio console, click the **Start** button to initiate the build of the console application and start it.
 15. The application will start a Command Prompt window. When prompted for password, type the password that you specified in the deployment in Exercise 1 to connect to Azure SQL Database.
 16. Leave the console app running and switch to the **SQL Management Studio** console.
 17. In the **Object Explorer** pane, right-click the **medical database** and, in the right-click menu, click **New Query**.
 18. From the query window, run the following query to verify that the data that loaded into the database from the console app is encrypted.

```
sqlTypeCopy
```

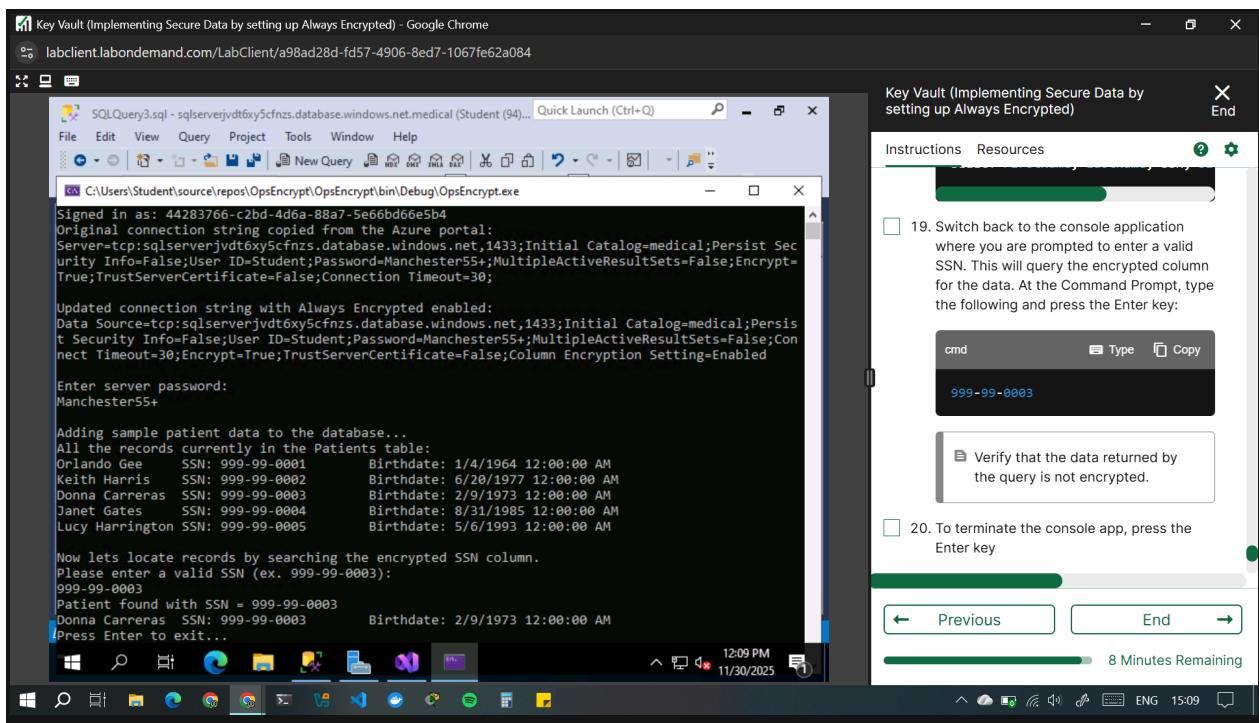
```
SELECT FirstName, LastName, SSN, BirthDate FROM Patients;
```



19. Switch back to the console application where you are prompted to enter a valid SSN. This will query the encrypted column for the data. At the Command Prompt, type the following and press the Enter key:

cmdTypeCopy

999-99-0003



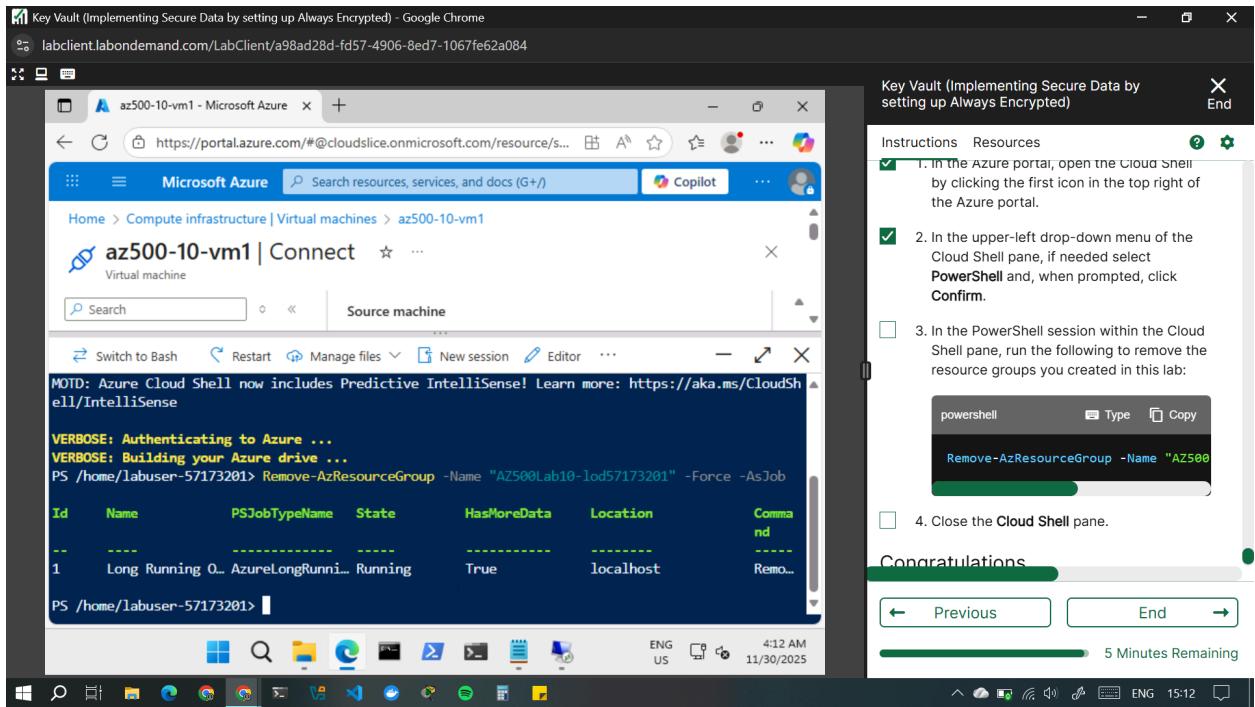
20. Verify that the data returned by the query is not encrypted.

21. To terminate the console app, press the Enter key

Clean up resources

Remember to remove any newly created Azure resources that you no longer use. Removing unused resources ensures you will not incur unexpected costs.

1. In the PowerShell session within the Cloud Shell pane, run the following to remove the resource groups you created in this lab:
2. powershellTypeCopy
3. `Remove-AzResourceGroup -Name "AZ500Lab10-1od57171145" -Force -AsJob`
4. Close the **Cloud Shell** pane.



Conclusion

This lab provided hands-on experience in implementing secure data practices using Azure Key Vault and Always Encrypted for Azure SQL Database. By deploying the foundational infrastructure, configuring Key Vault, and integrating it with a client application and SQL database, I gained practical understanding of how encryption keys and secrets can be centrally managed and securely consumed by applications. Implementing Always Encrypted demonstrated how sensitive data remains protected both at rest and in transit, with decryption only occurring within trusted client applications. The lab also reinforced key security concepts such as identity-based access control, application registration in Microsoft Entra ID, and securely retrieving credentials from Key Vault rather than embedding them in code. Overall, this exercise strengthened my ability to design and implement secure cloud architectures that safeguard sensitive information while maintaining application functionality and compliance with security best practices.