AZ-203 Developer Training

Implement Azure Security

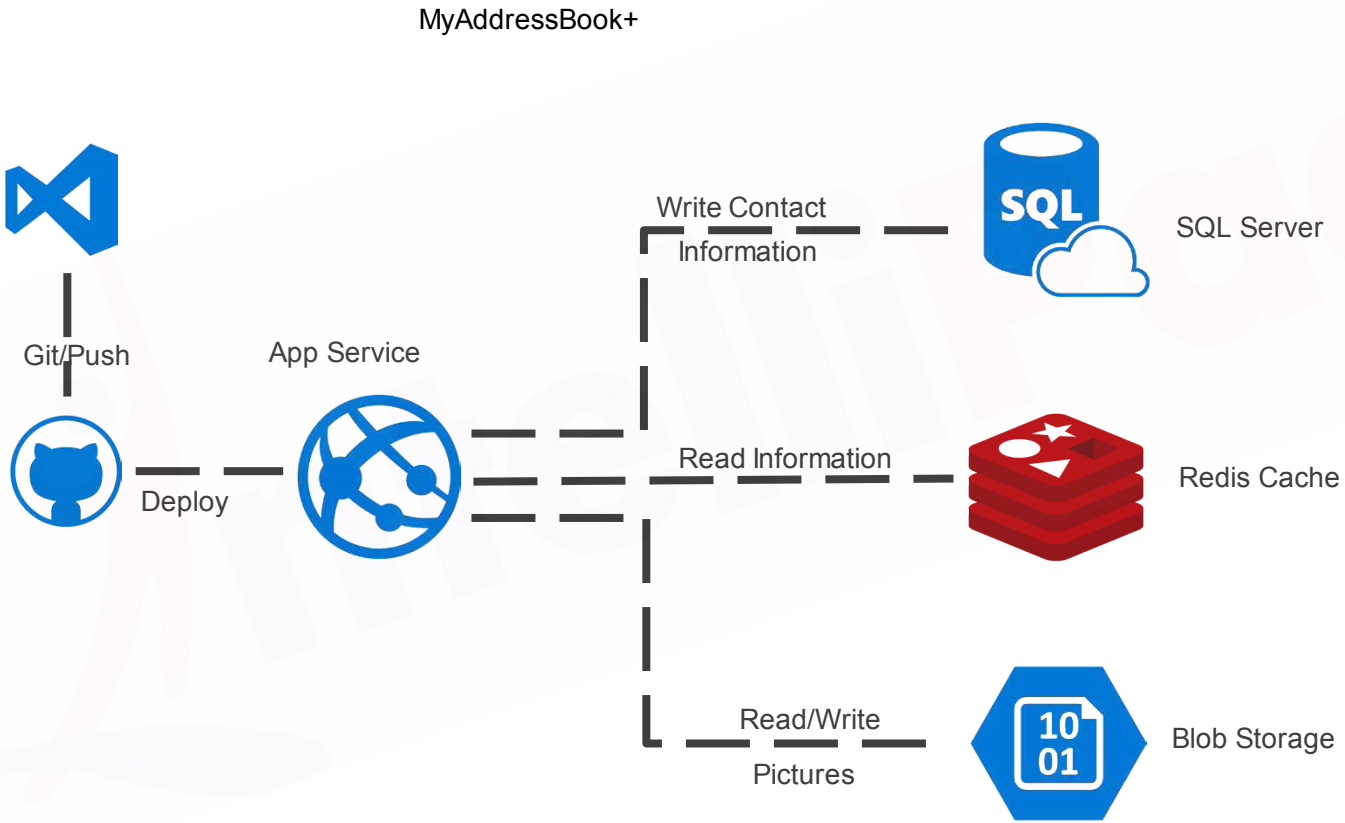
Agenda

- What to expect from this course?
- Getting familiar with our sample web application “MyAddressBook+”
- Demo
- Understanding weak security points in our application
- What Microsoft Azure has to offer? Module summary
- What are we trying to protect? keys vs. secrets
- Understanding Azure Key Vault and how it works
- Demo: Azure Key Vault
- Securing Azure SQL connection string using Managed Service Identity (MSI)
- Demo: Managed Service Identity (MSI)
- Microsoft tools to help you manage identity

Agenda

- Understanding Azure Storage Service Encryption for data at rest
- Configuring customer-managed keys (BYOK) for storage account
- Demo: Configuring customer-managed keys (BYOK) for MyAddressBook+
- Azure Disk Encryption for IaaS Virtual Machines
- Demo: Azure Disk Encryption Summary
- Understanding SSL/TLS
- Azure Portal & Power Shell
- Understanding Azure Key Vault

MyAddressBook+



Hands-on

Hands-on

- MyAddressBook+ in action
- Showing the code
- Examining the AppService in Microsoft Azure
- Pointing out the weak security points in this application

Let's Talk Security

Plain data in Azure SQL Server, plain SQL connection string

Plain Azure Redis Cache connection string

Contact pictures stored in Azure blob storage are encrypted, however, we prefer to bring and manage our own keys

Insecure communication via HTTP for custom domain names

Improving Security

Security Concern	Microsoft Azure Solution
Plain Azure Redis Cache connection string	Store the connection string in “Microsoft Azure Key Vault”
Images stored in Azure blob storage are encrypted at rest using Microsoft managed keys	Configure “Encryption at Rest” to use customer managed keys via Azure Key Vault
Plain data in Azure SQL Server/Connection string	Protect SQL Server data using “Always Encrypted”/MSI
Insecure communication via HTTP for custom domain names	Secure Azure App Service communications for custom domain names by configuring HTTPS

Microsoft Azure Key Vault

What Are We Trying to Protect?

IntelliPaat

Key

Cryptographic keys used in other Microsoft Azure services such as "Always Encrypted" or "data encryption at rest"

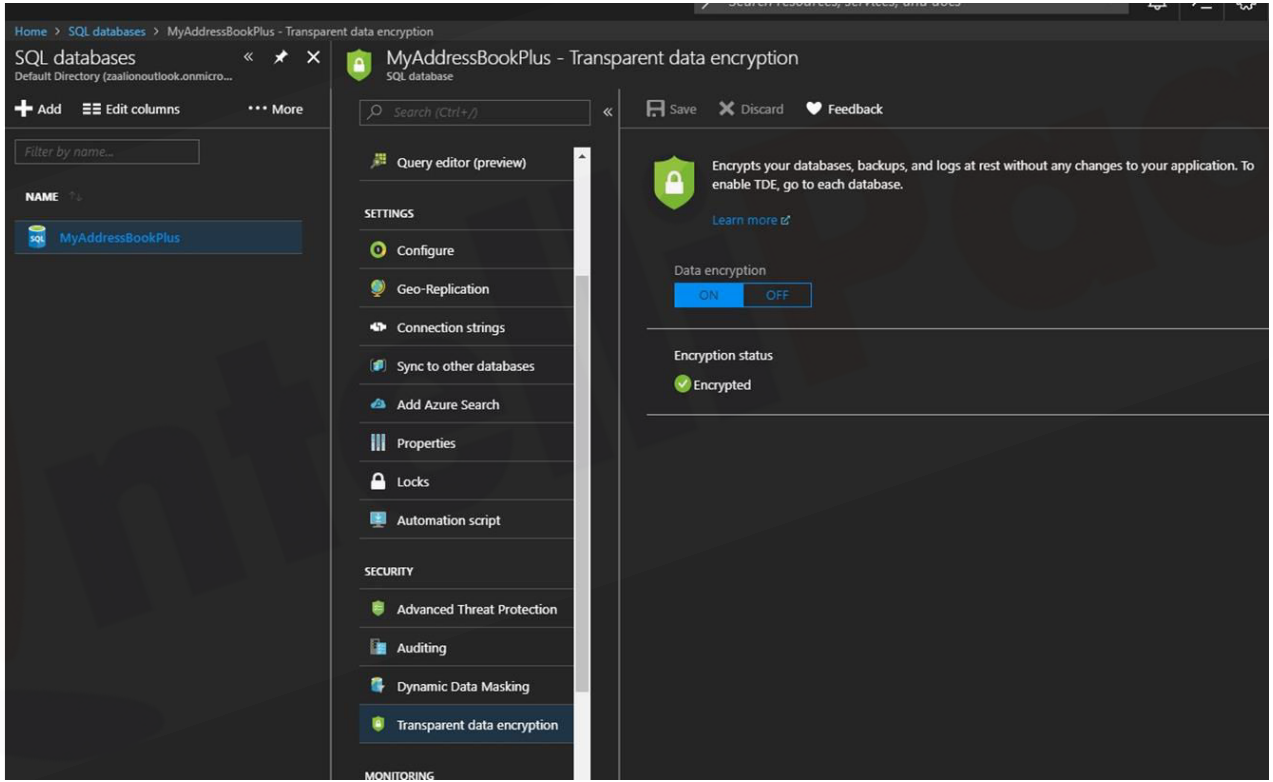
Secret

Any sensitive information including SQL server, Redis, storage connection strings or other information your application might need at runtime

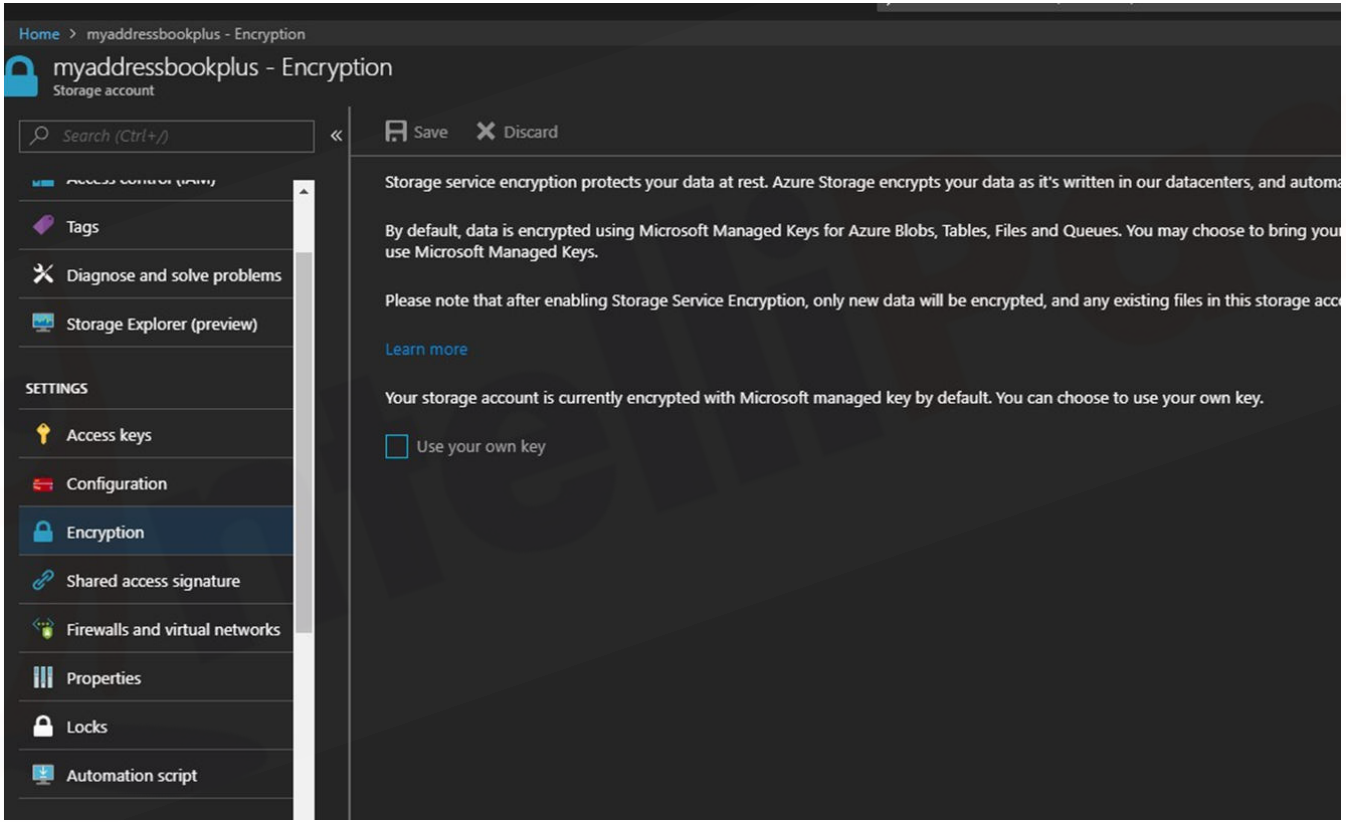
Certificate

x509 certificates being used in HTTPS/SSL communications

Keys



Keys



Secrets

```
<add key="CacheConnection"
value="myaddressbookplus.redis.cache.windows.net:6380,password=hQwiwqd+jij2nZZHzyW5AtawOTq71P4DkNn3n5BFPrw=,ssl=True,abortConnect=False"/>
```

- Azure Redis cache connection string

Secrets

```
<add key="StorageConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=myaddress
bookplus;AccountKey=BgAVowM+oErfnie9myvJ5XiBU0RAXtYlmyqMwEZ
ptz+pUaK2ERqZI1PJW1WL5vHofijj2SIYJq0eF7DE17OPVg==;EndpointS
uffix=core.windows.net"/>
```

- Azure Storage connection string

Certificates

- x509 certificates being used in HTTPS/SSL communications

What is Azure Key Vault?

What is Azure Key Vault?

What is Azure Key Vault?

Hands-on

Hands-on

- Create a new Vault in Azure Key Vault
- Move the Redis cache connection string to the new Vault (as a new secret)
- Register MyAddressBook+ with Azure Active Directory
- Configure MyAdressBook+ code
  - Remove Redis cache connection string from configuration
  - Add support to load the connection string from Azure Key Vault
- Confirm that MyAddressBook+ can use the cache

Code Changes

```
<!– web.config  -->
<add  key="ClientId"        value="686251ec-01b5-4877-9663-7bacf2d23bcc"              />
<add  key="ClientSecret“value="cQY9G2kEXr3/+8oqUYMT0IWYTXB9UWBJt8Ro0WMKJ48="          />
```

- Active directory Client Id, Client Secret

Code Changes

```
<!– web.config -->

<add key="ClientId" value="686251ec-01b5-4877-9663-7bacf2d23bcc" />

<add key="ClientSecret" value="cQY9G2kEXr3/+8oqUYMT0IWYTXB9UWBJt8Ro0WMKJ48=" />

<add key="CacheConnectionSecretUri"
value="https://myaddressbookplusvault.vault.azure.net:443/secrets/CacheConnection
/8233eacf6f97446a89aea139172dc616" />
```

- Azure Active directory (AAD) Client Id, Client Secret
- Redis cache connection string secret Key Vault URL

Code Changes

```
<!– Global.asax.cs -->

var kv = new KeyVaultClient(new
KeyVaultClient.AuthenticationCallback(KeyVaultService.GetToken));

var sec =
kv.GetSecretAsync(WebConfigurationManager.AppSettings["CacheConnectionSecretUri"])
.Result;

KeyVaultService.CacheConnection  = sec.Value;
```

- Read the secret from Azure Key Vault and save in memory.

Enable Key Vault Soft-delete


```
# Existing key vault

($resource = Get-AzureRmResource -ResourceId (Get-AzureRmKeyVault
-VaultName “MyAddressBookVault").ResourceId).Properties | Add-
Member -MemberType "NoteProperty" -Name "enableSoftDelete“ -Value
"true"

Set-AzureRmResource -ResourceId $resource.ResourceId -Properties
$resource.Properties
```

- Soft-delete Allows recovery of deleted vaults and vault objects including keys, secrets, and certificates

Use Key Vault Soft-delete

```
# New key vault

New-AzureRmKeyVault -VaultName "MyAddressBookVault" -ResourceGroupName
"MyRG" -Location "westus" -EnableSoftDelete
```

- Soft-delete Allows recovery of deleted vaults and vault objects including keys, secrets, and certificates

Enable Key Vault ”Do Not Purge”

```
# New key vault

New-AzureRmKeyVault -VaultName "MyAddressBookVault" -ResourceGroupName
“MyRG" -Location "westus" -EnableSoftDelete
```

- “Do Not Purge” prevents accidental purging of deleted vaults and vault objects including keys, secrets, and certificates.

Enable Key Vault ”Do Not Purge”

```
# Existing key vault

($resource = Get-AzureRmResource -ResourceId (Get-AzureRmKeyVault
-VaultName “MyAddressBookVault").ResourceId).Properties | Add-
Member-MemberType NoteProperty -Name enablePurgeProtection -
Value "true"

Set-AzureRmResource -ResourceId $resource.ResourceId -Properties
$resource.Properties
```

- “Do Not Purge” prevents accidental purging of deleted vaults and vault objects including keys, secrets, and certificates.

Set and retrieve a secret from Azure Key Vault using: Azure CLI

From Azure Key Vault using Azure CLI: Step 1

Open Azure Cloud Shell

Use any of the ways to open the Cloud Shell

- Select Try it in the upper-right corner of a code block.

- Open Cloud Shell in your browser.

- Select the Cloud Shell button on the menu in the upper-right corner of the Azure Portal

From Azure Key Vault using Azure CLI: Step 2

To sign in to Azure using the CLI you can type:

Azure CLI

```
az login
```

Create a resource group

A resource group is a logical container into which Azure resources are deployed and managed. The following example creates a resource group named *ContosoResourceGroup* in the *eastus* location.

Azure CLI

```
az group create --name "ContosoResourceGroup" --location eastus
```

From Azure Key Vault using Azure CLI: Step 3

Create a Key Vault

Next you will create a Key Vault in the resource group created in the previous step. You will need to provide some information:

- For this, we use **Contoso-vault2**. You must provide a unique name in your testing.
- Resource group name **ContosoResourceGroup**.
- The location **East US**.

Azure CLI

```
az keyvault create --name "Contoso-Vault2" --resource-group "ContosoResourceGroup" --location eastus
```

From Azure Key Vault using Azure CLI: Step 4

Add a secret to Key Vault

To add a secret to the vault, you just need to take a couple of additional steps. This password could be used by an application. The password will be called *ExamplePassword* and will store the value of hVFkk965BuUv in it.

Type the commands below to create a secret in Key Vault called *ExamplePassword* that will store the value hVFkk965BuUv :

Azure CLI

```
az keyvault secret set --vault-name "Contoso-Vault2" --name "ExamplePassword" --value "hVFkk965BuUv"
```

You can now reference this password that you added to Azure Key Vault by using its URI. Use *https://ContosoVault.vault.azure.net/secrets/ExamplePassword* to get the current version.

To view the value contained in the secret as plain text:

Azure CLI

```
az keyvault secret show --name "ExamplePassword" --vault-name "Contoso-Vault2"
```

From Azure Key Vault using Azure CLI: Last Step

You have created a Key Vault, stored a secret, and retrieved it.

Clean up resources

When no longer needed, you can use the *az group delete* command to remove the resource group, and all related resources. You can delete the resources as follows:

Azure CLI

```
az group delete --name ContosoResourceGroup
```

Set and retrieve a secret from Azure Key Vault using: Powershell

From Azure Key Vault using Powershell: Step 1

Open Azure Cloud Shell

Use any of the ways to open the Cloud Shell

- Select Try it in the upper-right corner of a code block.

- Open Cloud Shell in your browser.

- Select the Cloud Shell button on the menu in the upper-right corner of the Azure Portal

From Azure Key Vault using Powershell: Step 2

If you are running PowerShell locally, you also need to run *Login-AzAccount* to create a connection with Azure.

Azure CLI

```
Login-AzAccount
```

Create a resource group

Create an Azure resource group with *New-AzResourceGroup*. A resource group is a logical container into which Azure resources are deployed and managed.

Azure CLI

```
New-AzResourceGroup -Name ContosoResourceGroup -Location EastUS
```

From Azure Key Vault using Powershell: Step 3

Create a Key Vault

Next you create a Key Vault. When doing this step, you need some information:

Although we use "Contoso KeyVault2" as the name for our Key Vault throughout this quickstart, you must use a unique name.

- Vault name *Contoso-Vault2*.
- Resource group name *ContosoResourceGroup*.
- Location *East US*.

Azure CLI

```
New-AzKeyVault -Name 'Contoso-Vault2' -ResourceGroupName 'ContosoResourceGroup' -Location 'East US'
```

The output of this cmdlet shows properties of the newly created key vault. Take note of the two properties listed below:

- **Vault Name:** In the example that is Contoso-Vault2. You will use this name for other Key Vault cmdlets.
- **Vault URI:** In this example that is *https://contosokeyvault.vault.azure.net/*. Applications that use your vault through its REST API must use this URI.

From Azure Key Vault using Powershell: Step 3...

Continued....

After vault creation your Azure account is the only account allowed to do anything on this new vault.

```
Vault Name                      :
Resource Group Name             : ContosoResourceGroup
Location                        : East US
Resource ID                     : /subscriptions/                                  /resourceGroups/ContosoResourceGr
                                  oup/providers/Microsoft.KeyVault/vaults/
Vault URI                       : https://                 .vault.azure.net
Tenant ID                       :
SKU                             : Standard
Enabled For Deployment?         : False
Enabled For Template Deployment? : False
Enabled For Disk Encryption?    : False
Access Policies                 :
                                  Tenant ID                 :
                                  Object ID                 :
                                  Application ID            :
                                  Display Name              :
                                  Permissions to Keys       : get, create, delete, list, update, import, backup,
                                  restore
                                  Permissions to Secrets    : all
                                  Permissions to Certificates : all


Tags                            :
```

From Azure Key Vault using Powershell: Step 4

Add a secret to Key Vault

To add a secret to the vault, you just need to take a couple of additional steps. This password could be used by an application. The password will be called *ExamplePassword* and will store the value of hVFkk965BuUv in it.

First convert the value of **hVFkk965BuUv** to a secure string by typing:

Azure CLI

```
$secretvalue = ConvertTo-SecureString 'hVFkk965BuUv' -AsPlainText -Force
```

Then, type the PowerShell commands below to create a secret in Key Vault called ExamplePassword with the value hVFkk965BuUv :

Azure CLI

```
$secret = Set-AzKeyVaultSecret -VaultName 'ContosoKeyVault' -Name 'ExamplePassword' -SecretValue $secretvalue
```

To view the value contained in the secret as plain text:

Azure CLI

```
(Get-AzKeyVaultSecret -vaultName "Contosokeyvault" -name "ExamplePassword").SecretValueText
```

Now, you have created a Key Vault, stored a secret, and retrieved it.

From Azure Key Vault using Powershell: Last Step

Clean up resources

When no longer needed, you can use the *Remove-AzResourceGroup* command to remove the resource group, Key Vault, and all related resources.

Azure CLI

```
Remove-AzResourceGroup -Name ContosoResourceGroup
```

Then, type the PowerShell commands below to create a secret in Key Vault called ExamplePassword with the value hVFkk965BuUv :

Azure CLI

```
$secret = Set-AzKeyVaultSecret -VaultName 'ContosoKeyVault' -Name 'ExamplePassword' -SecretValue $secretvalue
```

To view the value contained in the secret as plain text:

Azure CLI

```
(Get-AzKeyVaultSecret -vaultName "Contosokeyvault" -name "ExamplePassword").SecretValueText
```

Now, you have created a Key Vault, stored a secret, and retrieved it.

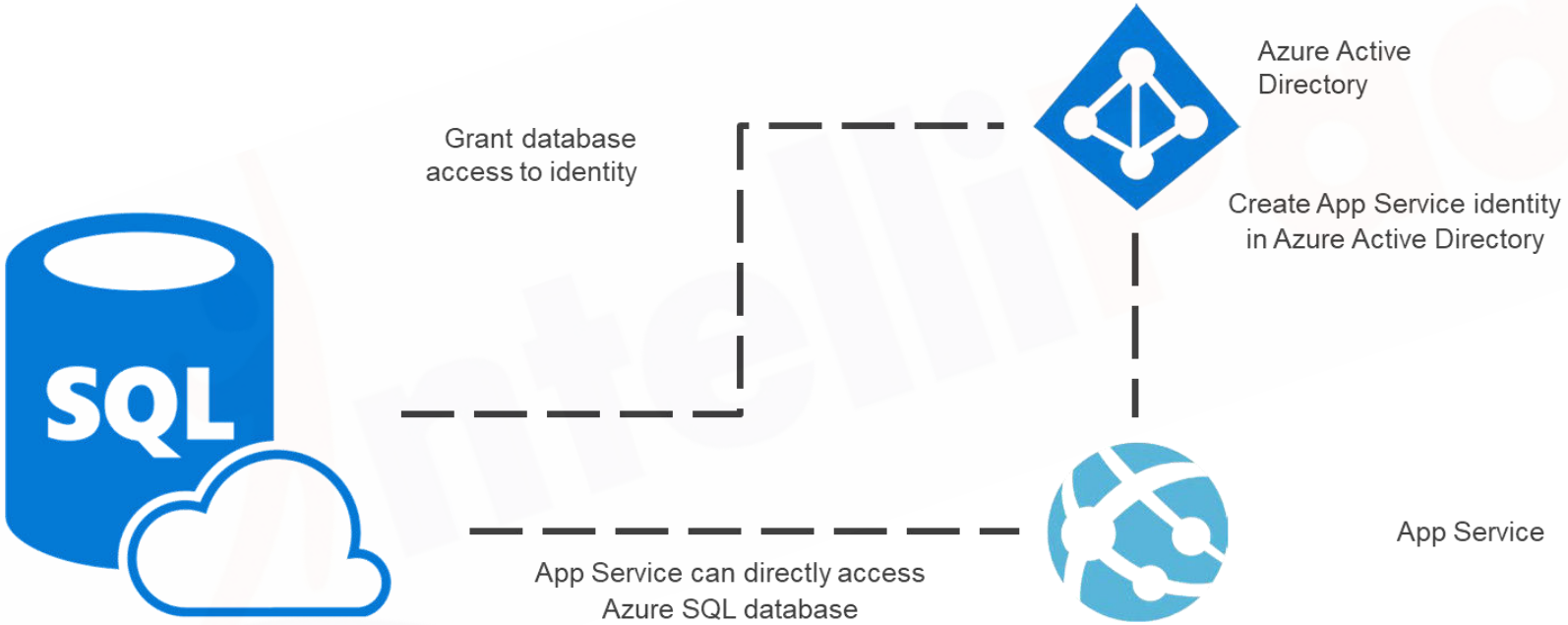
Hands-on

- Checking if soft-delete is enabled for our Vault
- Enabling soft-delete for the Key Vault
- Verifying that soft-delete is indeed enabled
- Deleting a vault protected by soft-delete
- Recovering the deleted key vault
- Purging a key vault

Managed Service Identity (MSI)

What is Managed Service Identity (MSI)?

What is Managed Service Identity (MSI)?



Why Managed Service Identity Is Recommended?

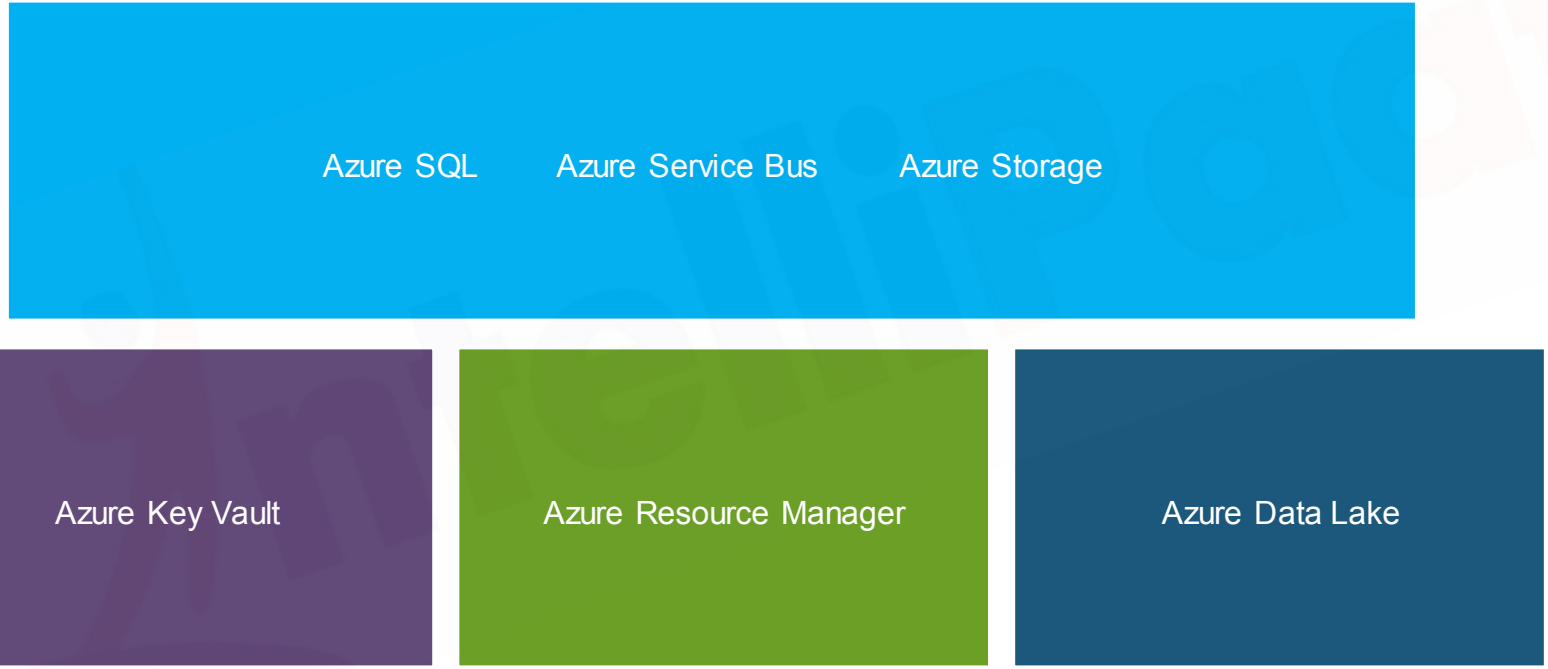
No need to authenticate to Azure Key Vault to get secrets

No client id and client secret is needed in the code

Easier to configure comparing to Azure Key Vault

Easier to configure comparing to Azure Key Vault

Services That Support Managed Service Identity



Secrets

```
<add name="SqlDataConnection" connectionString="data
source=zaalion.database.windows.net;initial
catalog=MyAddressBookPlus;persist security info=True;user
id=AppServiceLogin;password=P@$$w0rd;MultipleActiveResultSe
ts=True;" />
```

- Azure SQL database connection string

Hands-on

- Enable Managed Service Identity for MyAddressBook+
- Configure Azure SQL Database to grant access to the new identity
- Update MyAddressBook+ code
  - Remove credentials (username, password) from connection string
  - Modify ASP.NET code
- Verify MyAddressBook+ works as expected

Code Changes

```
<!– web.config –>

<!– before –>

<add name="SqlDataConnection" connectionString="data
source=zaalion.database.windows.net;initial catalog=MyAddressBookPlus;persist security
info=True;user id=AppServiceLogin;password=P@$$w0rd;MultipleActiveResultSets=True;" />

<!– after –>

<add name="SqlDataConnection" connectionString="data
source=zaalion.database.windows.net;initial catalog=MyAddressBookPlus;persist security
info=True;MultipleActiveResultSets=True;" />
```

- Remove user id and password from SqlDataConnection

Code Changes

```
Install-Package Microsoft.Azure.Services.AppAuthentication -ProjectName
MyAddressBookPlus
```

- Install the Microsoft.Azure.Services.AppAuthentication Nuget package

Code Changes

```
<!– ContactRepository -->

var accesstoken = (new
AzureServiceTokenProvider()).GetAccessTokenAsync("https://database.windows.net/").Result;

db = new SqlConnection()            {

     AccessToken = accesstoken,

     ConnectionString = connectionstring

};
```

- Update SqlConnection to use AAD access token for authentication

Useful Tools from Microsoft

Azure Services Authentication Extension for Visual Studio 2017 update 5.
Allows projects that use the Microsoft.Azure.Services.AppAuthentication library to access Azure resources using their Visual Studio accounts.

Microsoft Credential Scanner (preview)
Monitors all incoming commits on GitHub and checks for specific Azure tenant secrets such as Azure SQL connection strings.

Hands-on

- Using Azure Services Authentication Extension

Azure Storage Service Encryption for Data at Rest

Data in Transit vs. Data at Rest

IntelliPaat

Data in transit

When data is being transferred between components, locations, or programs, such as over the network, across a service bus

Data at rest

Inactive data that is stored physically in any digital form (e.g. databases, files, data warehouses)

Azure Storage Service Encryption for Data at Rest

Organizational security

Your security strategy requires all data at rest to be encrypted at all times

Compliance commitments

Your organization is required by customers, partners, or government regulations to encrypt data at rest

Azure Storage Supported Types

Azure Blob storage

Azure Table storage

Azure Files

Azure Queue storage

Azure Managed Disks

Azure Storage Service Encryption for Data at Rest

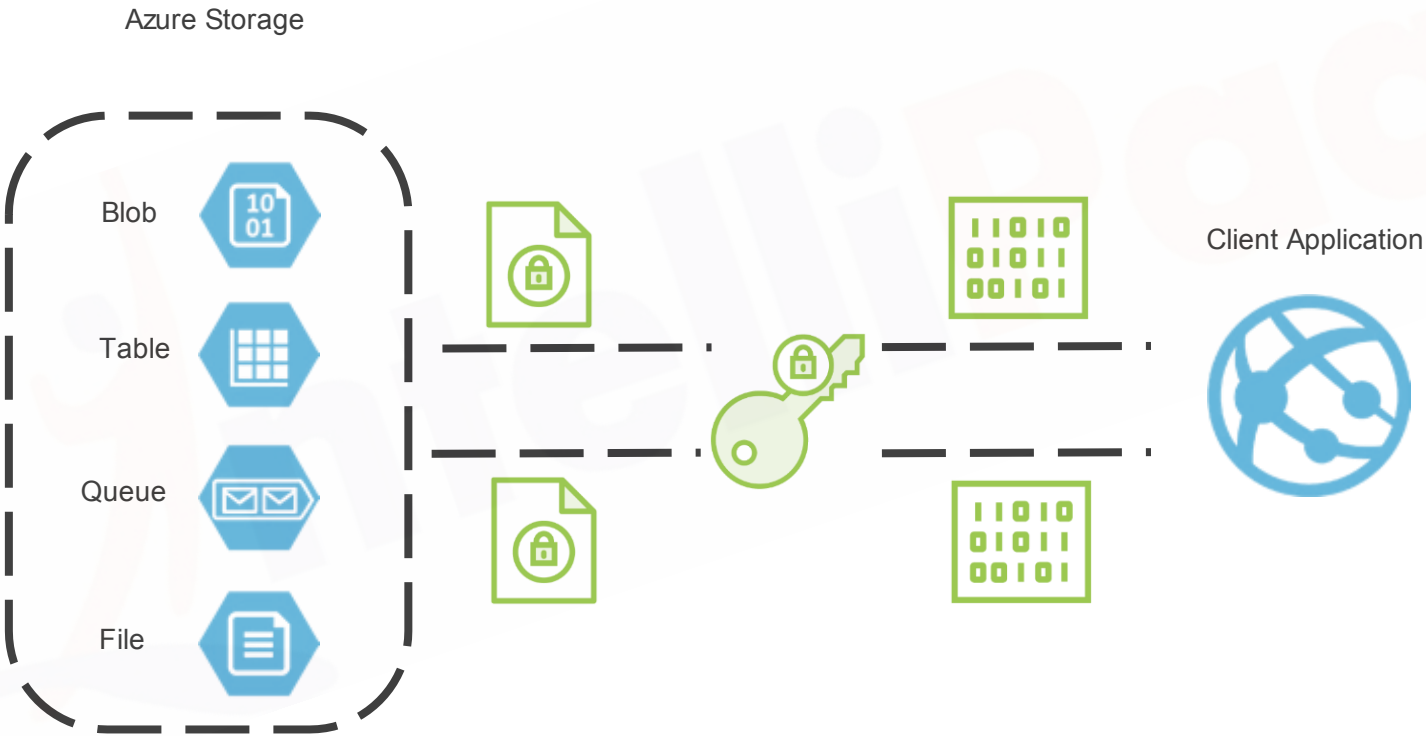
Storage Service Encryption (SSE) is enabled for all new and existing storage accounts and cannot be disabled

Your data is secured by default, you don't need to modify your code or applications to take advantage of Storage Service Encryption

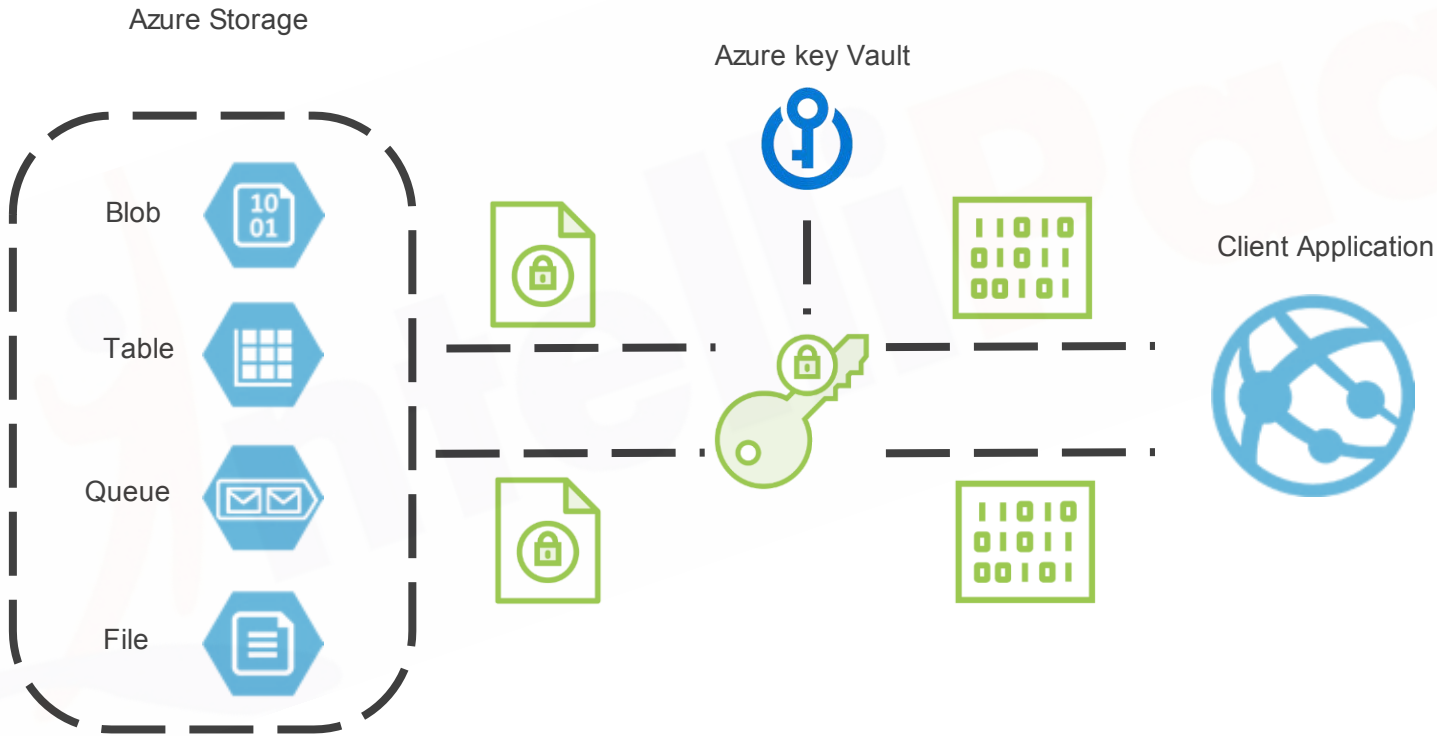
SSE automatically encrypts data in all performance tiers (Standard and Premium), all deployment models (Azure Resource Manager and Classic)

Azure storage platform is encrypted through *256-bit AES encryption*, one of the strongest block ciphers available

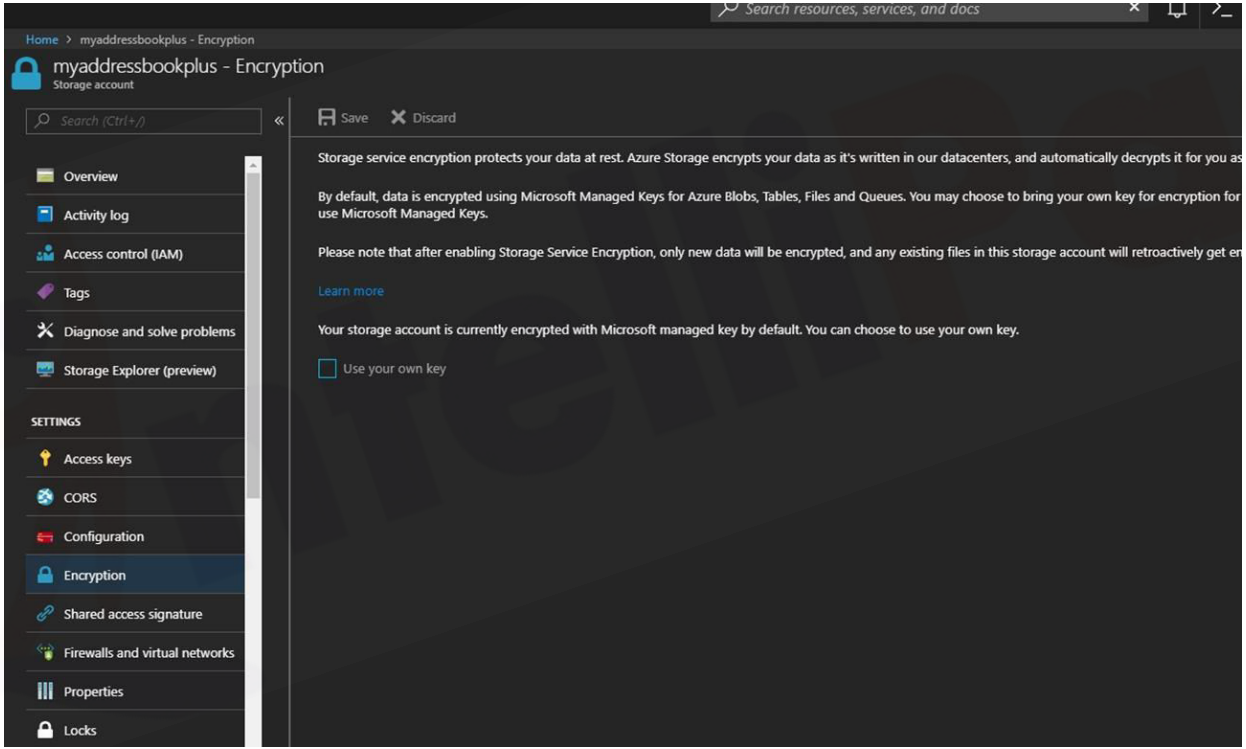
How Does Encryption for Data at Rest Work?



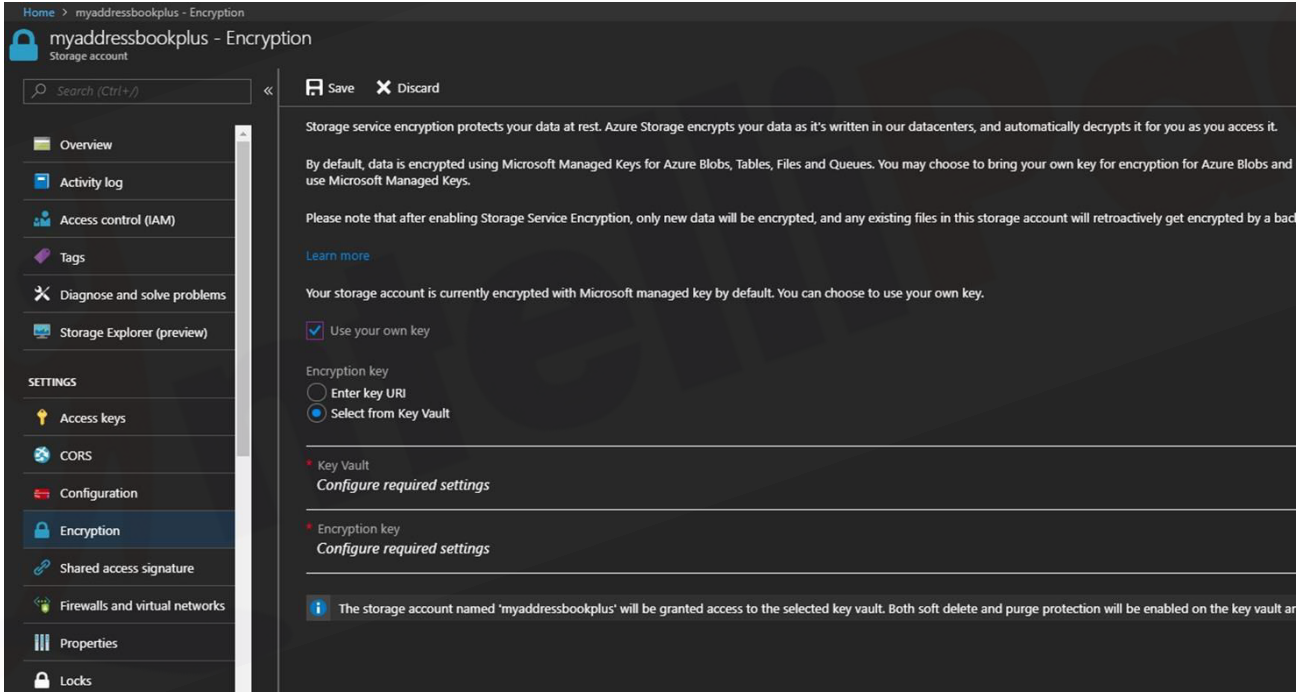
Customer-managed Keys (BYOK) for Storage Account



Customer-managed Keys (BYOK) for Storage Account



Customer-managed Keys (BYOK) for Storage Account



Using Customer Managed Keys with SSE

The storage account and the key vault must be in the same region

Two key protection features, *Soft Delete* and *Do Not Purge,* must also be enabled. These settings ensure the keys cannot be accidentally or intentionally deleted

SSE is available for Azure Managed Disks with Microsoft-managed keys, but not with customer managed keys. In lieu of Managed Disks supporting SSE with customer-managed keys, *Microsoft* recommends Azure Disk Encryption

Hands-on

Hands-on

- Configuring MyAddressBook+ storage account to use customer-managed keys for encryption at rest

Associate a Key with an Existing Storage

```
Set-AzureRmStorageAccount -ResourceGroupName
$storageAccount.ResourceGroupName -AccountName
$storageAccount.StorageAccountName -KeyvaultEncryption
-KeyName $key.Name -KeyVersion $key.Version -KeyVaultUri
$keyVault.VaultUri
```

You Are Already Using Disk Encryption!

Windows

BitLocker Drive Encryption is a data protection feature that addresses the threats of data theft or exposure from lost, stolen, or inappropriately decommissioned computers

Linux

"dm-crypt is a transparent disk encryption subsystem in Linux kernel versions 2.6 and later."

Wikipedia

Azure Disk Encryption for IaaS VMs

Defense in Depth

Multiple layers of security defense

Not Enabled by Default

Should specifically get enabled

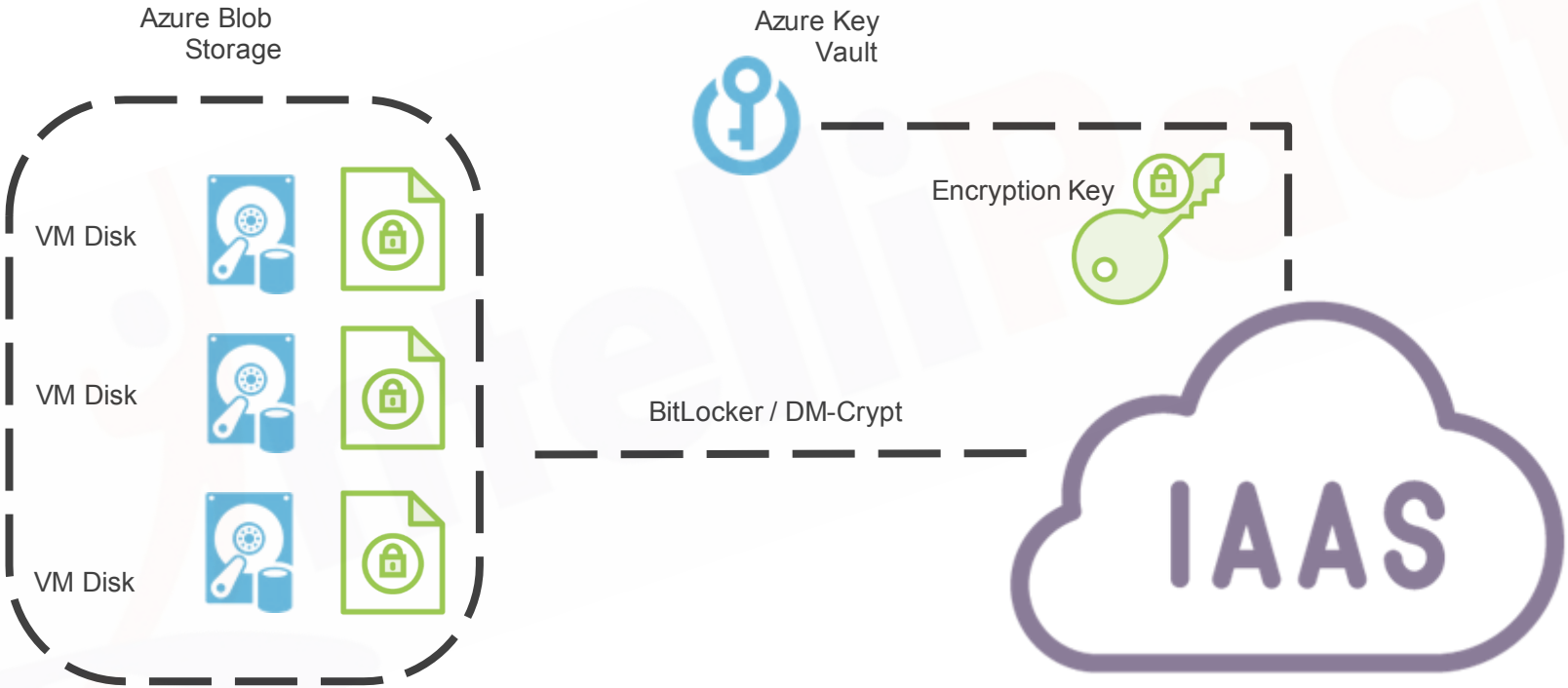
Azure Disk Encryption (ADE)

Helps you encrypt your IaaS virtual machine disks

ADE leverages BitLocker of Windows and the DM-Crypt of Linux

Is integrated with Azure Key Vault to help you manage the disk-encryption keys

How Does Azure Disk Encryption Work?



Hands-on

- Create a new Windows VM
  - Configure Azure Disk Encryption for the VM
  - Create an Azure Key Vault
  - Store an encryption key in the vault
  - Set the correct access to the key
  - Enable encryption option on the VM using Azure PowerShell
  - Verify that Disk Encryption is enabled
- Disable the encryption

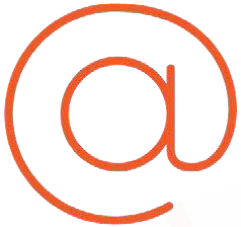
Encrypt a Running VM Using a Client Secret

```
Set-AzureRmVMDiskEncryptionExtension -ResourceGroupName
'MySecureGroupName' -VMName $vmName -AadClientID
$aadClientID -AadClientSecret $aadClientSecret -
DiskEncryptionKeyVaultUrl $diskEncryptionKeyVaultUrl -
DiskEncryptionKeyVaultId $KeyVaultResourceId;
```

Verify the Disks Are Encrypted

```
Get-AzureRmVmDiskEncryptionStatus -ResourceGroupName
'MySecureGroupName' -VMName 'MySecureVMName'
```

Sensitive Information



Email address

Date of birth

Phone number

Social security number

Salary

Password

Why Encrypt Data in Azure SQL Database?

Bad Guys

Hackers Could access your database files, then they can see confidential information

Good Guys

DBAs have full access over your database. Should they see your information?

Azure SQL Database “Always Encrypted”

Data encryption technology available in Azure SQL Database and SQL Server

Protect sensitive data at rest on the server

During movement between client and server

Ensuring that sensitive data never appears as plaintext inside the database system

Who Can See the Data Then?

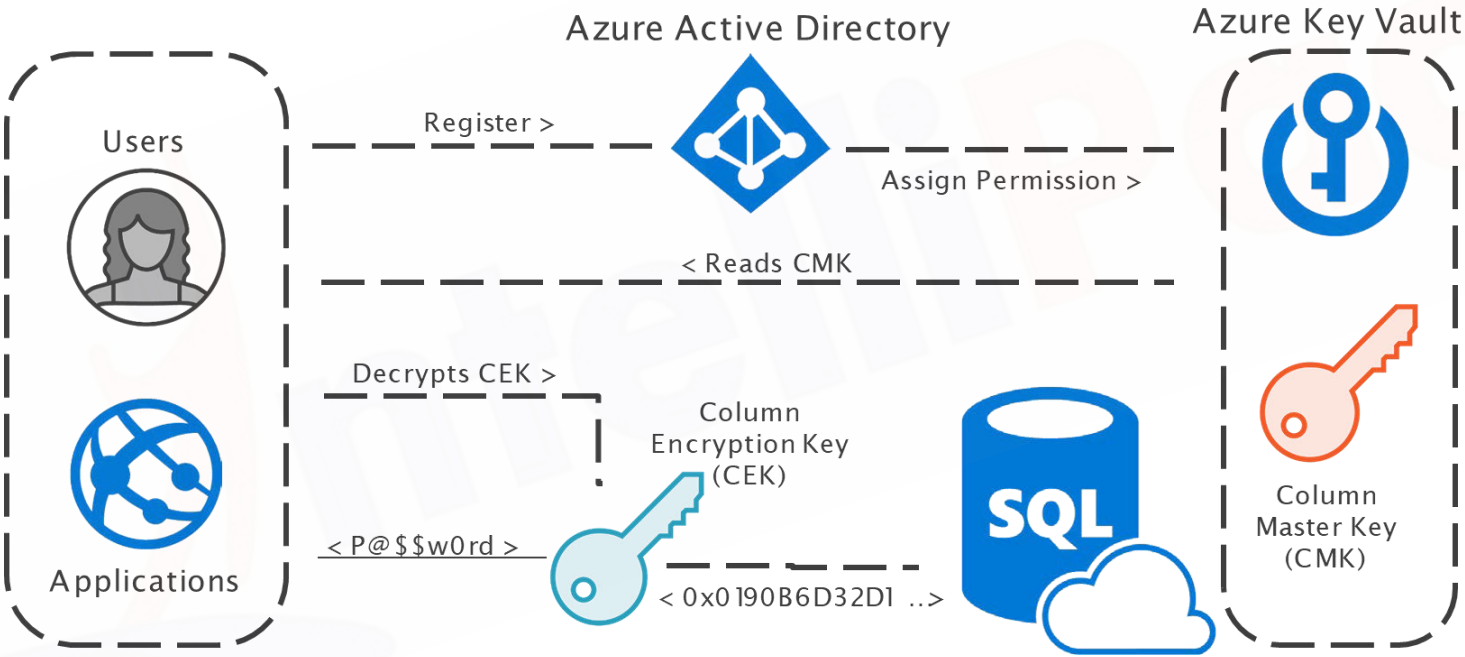
Users/Accounts

Who have the encryption key

Applications/Services

Which have the encryption key

How Does “Always Encrypted” Work?



Column Encryption Types

Randomized	Deterministic
Generates different encrypted value for the same plain text	Generates the same encrypted value for any given plain text
More secure, the encrypted values are difficult to guess	Easy to guess specially for small set of possible encrypted values
Prevents searching, grouping, indexing, and joining on encrypted columns	Allows lookups, equality joins, grouping and indexing on encrypted columns
Confidential comments, not searched	Government ID number, emails, etc.

Hands-on

Hands-on

- Add a SIN number column to the contacts table
- Configure Always Encrypted for the new column
  - Storing CMK in Azure Key Vault
- Update MyAddressBook+ code to work with the new updates
- Randomized vs. deterministic encryption in action
- Confirm that MyAddressBook+ can encrypt and decrypt the data

Code Changes for Always Encrypted

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

Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory
```

Code Changes for Always Encrypted

Column Encryption Setting=Enabled

- Enable “Always Encrypted” in the connection string

Code Changes for Always Encrypted

```
static void InitializeAzureKeyVaultProvider()

public async static Task<string> GetToken(string authority, string
resource, string scope)
```

- Register the Azure Key Vault provider with ADO.NET, so the CMK can be read from Key Vault at runtime

Code Changes for Always Encrypted

```
DynamicParameters parameter = new DynamicParameters();

parameter.Add("@SIN_Number", contact.SIN_Number, DbType.String,
ParameterDirection.Input, 9);
```

- Use query parameters with fixed length in your queries

Other Azure SQL Encryption Options

Always Encrypted

Client Side, data is "always encrypted" in transition & in the SQL database

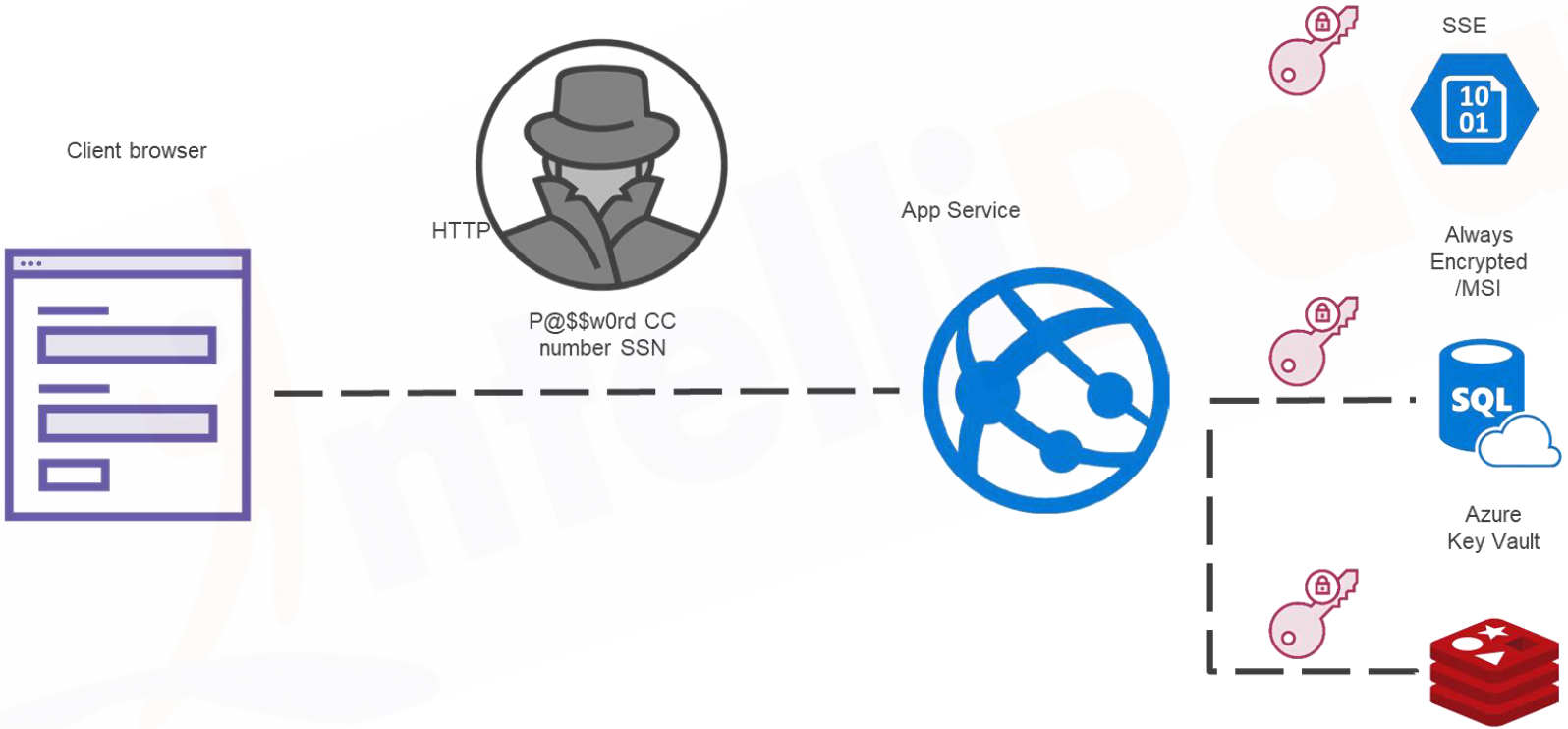
Transparent Data Encryption (TDE)

Server side, encrypts SQL Server, Azure SQL Database, and Azure SQL Data Warehouse data files (at rest)

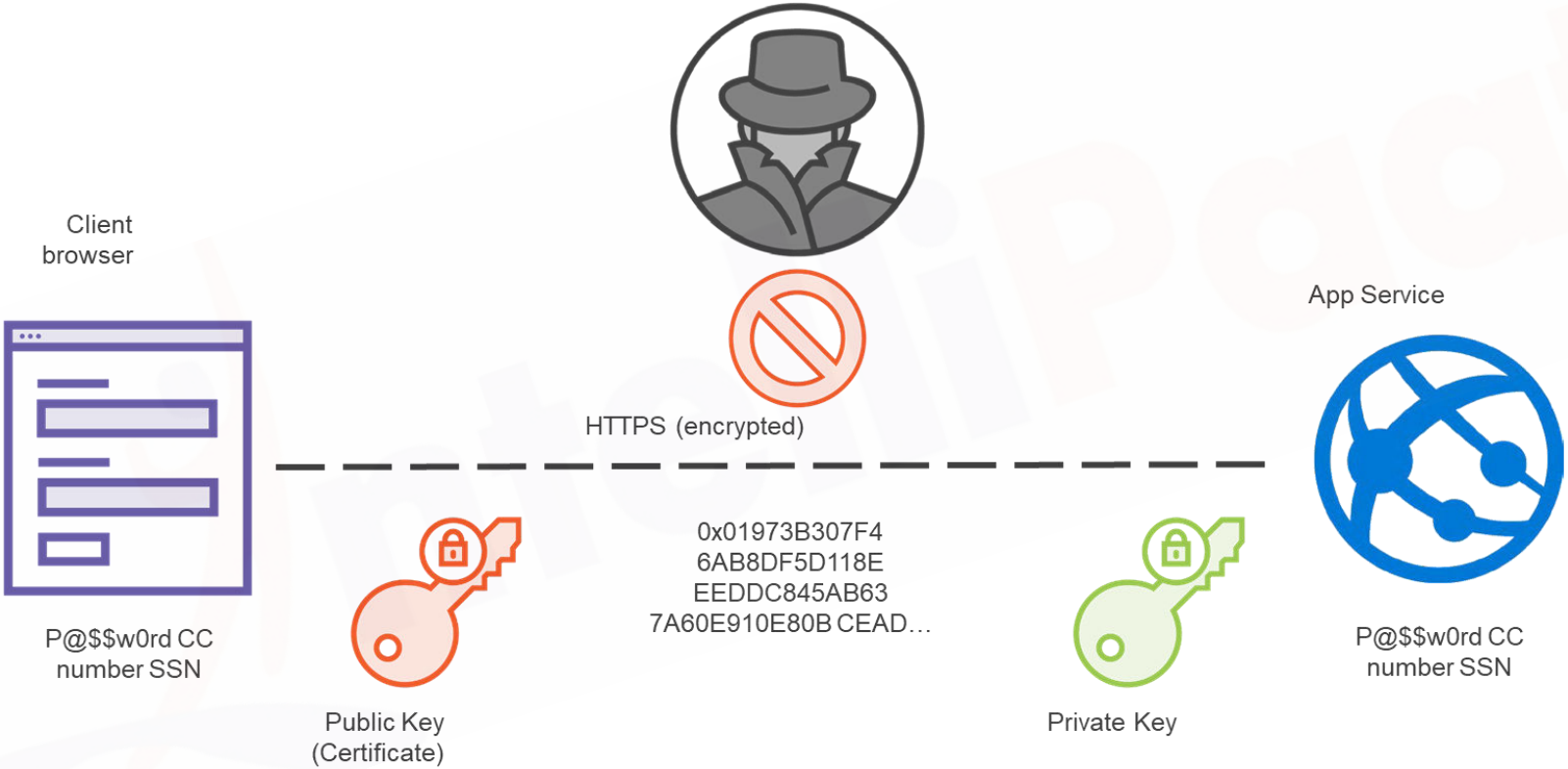
Hands-on

- Examining Transparent Data Encryption (TDE) option on the server level
- Examining Transparent Data Encryption (TDE) option on the database level

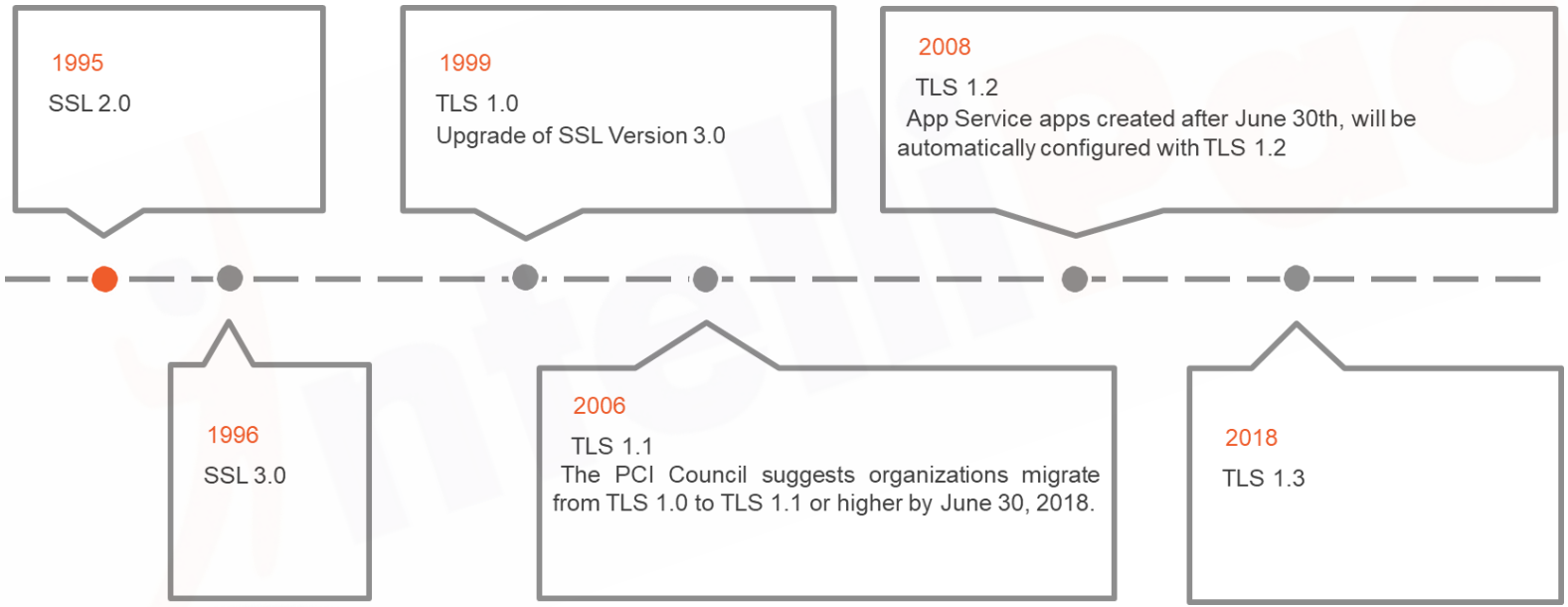
Securing Client-Server Communications



Understanding SSL/TLS



SSL/TLS



Configuring SSL for Azure App Service

Configure your App Service to use a custom domain name

Purchase the new SSL certificate for the custom domain name in Azure Portal (can bring your certificate)

Upload the new certificate to the Azure Key Vault

Create SSL bindings in the App Service and enable HTTPS only

Hands-on

Hands-on

- Purchase an SSL certificate for our custom domain
- Configuring SSL on MyAddressBook+ App Service
  - In Azure Portal
  - Using PowerShell
- “HTTPS Only” option
- Redirecting HTTP requests to HTTPS

Configuring SSL for Custom Domain

```
Set-AzureRmWebApp -Name $webappname -ResourceGroupName
$webappname -HostNames
@($cusdn,"$appname.azurewebsites.net")
```

- Add a custom domain to the App Service

Configuring SSL for Custom Domain

```
New-AzureRmWebAppSSLBinding -WebAppName $webappname -
ResourceGroupName $webappname-Name $cusdn -
CertificateFilePath $pfxPath -CertificatePassword
$certPassword -SslState SniEnabled
```

- Bind the SSL certificate to the App Service.

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor