In this lab, you will learn how to do the following:

Log into the Azure CLI
Understand Azure Storage encryption at rest
Configure Azure Storage CMK encryption using Azure CLI
Confirm the encryption settings using Azure CLI
Check Azure Storage encryption settings in the portal
Clean up

Azure Key Vault is a cloud-based service that allows you to store and manage cryptographic keys, secrets, and certificates. When you enable purge protection for an Azure Key Vault, it adds an extra layer of security to the key vault by preventing permanent deletion of any item in the key vault.

By default, when you delete an item in a Key Vault, it is moved to the soft delete state where it can be recovered for a certain period of time before it is permanently deleted. With purge protection enabled, you cannot permanently delete any item from the Key Vault, even after the retention period has expired.

Enabling purge protection is a best practice for ensuring that sensitive data is not accidentally or maliciously deleted from your Key Vault. It provides an additional layer of security and helps you comply with regulatory requirements.

To enable purge protection for an Azure Key Vault, you can use the Azure Portal, Azure PowerShell, or Azure CLI.

To create a new Azure storage account:
az storage account create -n $storageAccountName --resource-group $resource --sku Standard_LRS

To create a new Azure key vault with purge protection enabled:
az keyvault create --location eastus --name $keyVaultName --resource-group $resource --enable-purge-protection true

To create a new encryption key in the above key vault:
az keyvault key create --vault-name $keyVaultName --curve "P-256" --kty "RSA" --name "myStorageKey"

--curve: This is an optional parameter that specifies the elliptic curve to use for the key. "P-256" is a value for this parameter that specifies to use the NIST P-256 curve.
--kty: This is an optional parameter that specifies the key type. "RSA" is a value for this parameter that specifies to use the RSA algorithm for the key.

Important: By passing the --default-action Deny parameter, we configured the above storage account to deny all traffic if no firewall rule matches the incoming request. Make sure you use the az storage account update command to set this flag for existing storage accounts.

az storage account update -g $resource -n $storageAccountName --default-action Deny

In this lab you will use the key stored in your Azure key vault to configure customer-managed key encryption for an Azure storage account. Let's get started.

Understand Azure Storage Encryption at Rest
In this lab you will learn about customer-managed keys (CMK) for Azure Storage encryption at rest.

Azure storage account data is encrypted at rest by default and you cannot turn this feature off. However, by default, the encryption key is managed and stored by Microsoft Azure. In some cases (for instance, to meet compliance and regulatory requirements), you might decide to be in change of the storage encryption keys. In this case, Azure storage account CMK encryption is for you.

Follow these steps to use your own encryption key for Azure storage accounts:

Create an encryption key in the Azure key vault.

Configure the Azure storage account to use CMK encryption.
We already created a new encryption key for you in this lab.

Important: The purge protection feature should be enabled on your key vault.

In the next step, we will use Azure CLI to configure CMK for your storage account.

Configure Azure Storage CMK Encryption using Azure CLI
Before we can configure CMK, we need to configure the key vault to trust our Azure storage account, so the said storage account identity can read the encryption key.

First, use the following command to assign a managed system-assigned identity to your storage account:
az storage account update --resource-group $resource --name $storageAccountName --assign-identity

--assign-identity: This is an optional parameter that enables an Azure managed identity for the storage account.

Enabling an Azure managed identity for a storage account allows applications and services to authenticate to the storage account using Azure AD credentials, without having to manage and store any storage account access keys. This can improve security and simplify access management for the storage account. Once the command completes successfully, the specified storage account will have an Azure managed identity enabled.

Then store the new identity ID in a variable as follows:

storageIdentity=$(az storage account show --name $storageAccountName --query "identity.principalId" --output tsv)

Now you can configure a new Azure Key Vault access policy to allow the storage account identity to read the encryption keys:

az keyvault set-policy --name $keyVaultName --object-id $storageIdentity --key-permissions get unwrapKey wrapKey

The Azure CLI command you provided sets an access policy for a specified Azure Key Vault, granting the specified managed identity ($storageIdentity) permissions to perform specific operations on keys in the Key Vault.

--object-id: This is a required parameter that specifies the principal ID of the object to grant permissions to
--key-permissions: This is an optional parameter that specifies the permissions to grant on keys in the Key Vault. In this case, get unwrapKey wrapKey specifies that the managed identity can perform operations to get, unwrap, and wrap keys in the Key Vault.

At this point, you are ready to configure CMK for your storage account. First, store the key vault URI in a variable so the storage account can reference it:

keyvaultURI=$(az keyvault show --resource-group $resource --name $keyVaultName --query properties.vaultUri --output tsv)

Finally, use the following command to configure CMK for your storage account:

az storage account update --resource-group $resource --name $storageAccountName --encryption-key-source Microsoft.Keyvault --encryption-services blob --encryption-key-vault $keyvaultURI --encryption-key-name myStorageKey

The Azure CLI command you provided updates an Azure Storage account to enable client-side encryption for blob data using a key stored in an Azure Key Vault.

--encryption-key-source: This is a required parameter that specifies the source of the encryption key. Microsoft.Keyvault specifies that the key will be stored in an Azure Key Vault.
--encryption-services: This is an optional parameter that specifies the type of data to be encrypted. blob specifies that blob data will be encrypted.
--encryption-key-vault: This is a required parameter that specifies the URI of the Key Vault containing the encryption key. $keyvaultURI is a variable that should be replaced with the URI of the Key Vault.
--encryption-key-name: This is a required parameter that specifies the name of the encryption key within the Key Vault. myStorageKey is the name of the key.

When this command is executed, the Azure CLI updates the specified Storage account to enable client-side encryption for blob data using a key stored in the specified Azure Key Vault. The Storage account will now use the specified encryption key to encrypt and decrypt blob data.

--resource-group: Parent resource group for the storage account
--name: Storage account name
--encryption-key-source: Set to Microsoft.Keyvault, indicating that the encryption key should be read from the Azure key vault
--encryption-services: Specifies which Azure storage account service(s) to encrypt; accepted values are blob, file, queue, and table
--encryption-key-vault: Azure key vault address/URI
--encryption-key-name: The encryption key name

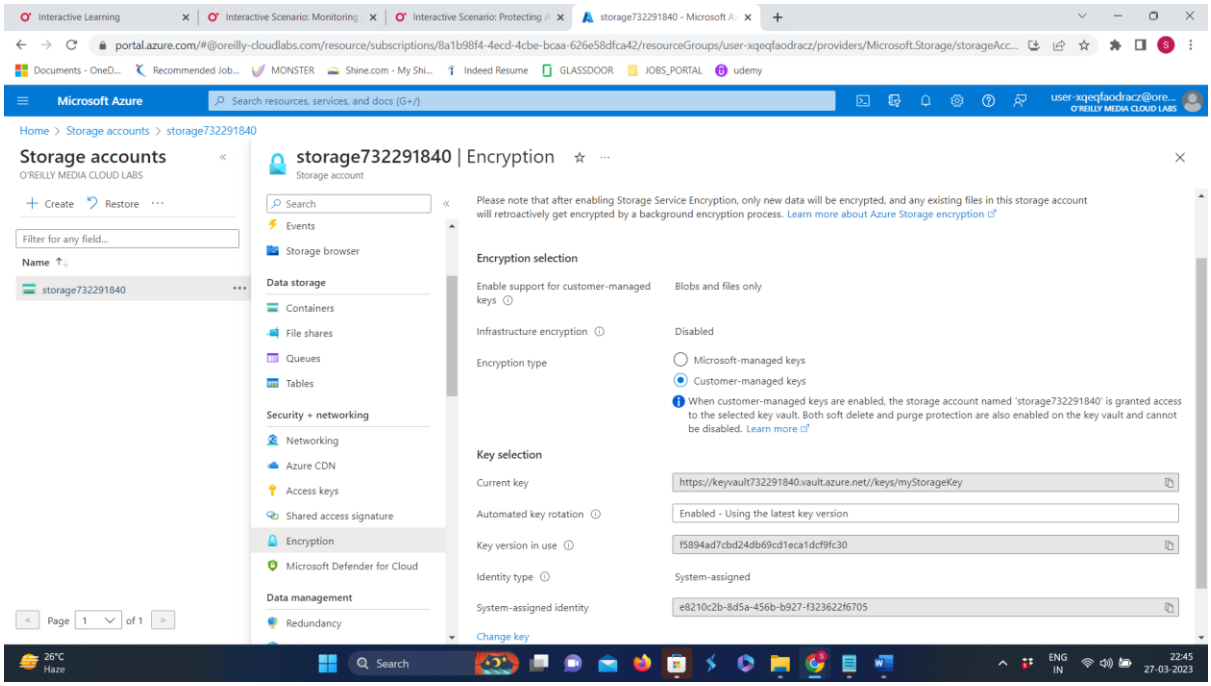
In the next step, you will use Azure CLI to check the Azure storage account encryption settings.

Confirm Encryption Settings using Azure CLI
Ue the following command to check the storage account encryption settings:

az storage account show --name $storageAccountName --query encryption.keyVaultProperties

Check Azure Storage Encryption Settings in the Azure Portal

Confirm that Encryption type is set to Customer-managed keys .
Confirm that you can see your key vault key URL in front of Current key.



Clean Up

Use the following command to clean up the storage accounts from your allocated resource group:

```
az storage account delete -n $storageAccountName -g $resource
```

Use the following command to list all storage accounts in your allocated resource group:

```
az storage account list -g $resource
```