In this lab, you will learn how to provision a new Azure Key Vault using the Azure CLI.

Azure Key Vault is an Azure service enabling you to securely store secrets, such as database passwords, connection strings, and more. These secrets can be accessed in a secure manner by your applications at runtime. You can also store other entities such as encryption keys, and certificates in a Key Vault. We will discuss secrets, keys, and certificates in these labs.

Log into the Azure CLI
Understand the Azure Key Vault
Provision an Azure Key Vault using the CLI
Check the new Key Vault in Azure portal
Clean up Azure Key Vault

Understaning Azure Key Vault
So what is an Azure Key Vault, and why should you use it in your Azure solutions?

Imagine you are developing a web application to be hosted in Azure. This web application needs a database connection string to talk to an Azure SQL Database. Where would you store the connection string? Would you hardcode it in the application code? Or put it in the application configuration? None of the mentioned approaches are recommended because the connection string might accidentally be checked into the source control, allowing anyone with access to the source to connect to the database. In other words, we don't want our database password exposed if the application code is compromised. In some cases, we don't even want the developers to see the database password. So what is the recommended approach? Microsoft Azure recommends using Azure Key Vaults.

You simply store your database connection string as a Key Vault secret. Then, you configure the Key Vault, allowing access to your application. At runtime, your application will connect to the Key Vault, and fetch the database connection string. No need to hardcode it in the code anymore.

You can store three different entities in the Key Vault, and they are:

Secrets: Entities such as passwords, database connection strings, or any custom secret you might need in your applications.
Keys: These are cryptographic keys, which can be used in your applications or by other Azure services to encrypt an entity. For example, Azure Disks can use a Key Vault Key to encrypt a VM (virtual machine) disk. Other important use cases are storing customer managed keys (CMKs) for Cosmos DB, Storage Accounts, and more.
Certificates: These include custom X.509 SSL certificates you use to secure your custom domains. These certificates can be imported to the Key Vault and used or downloaded when needed.
In this lab, we will create an Azure Key Vault using the Azure CLI, confirm it's created using both the CLI and the Azure portal, and finally we will clean it up at the end.

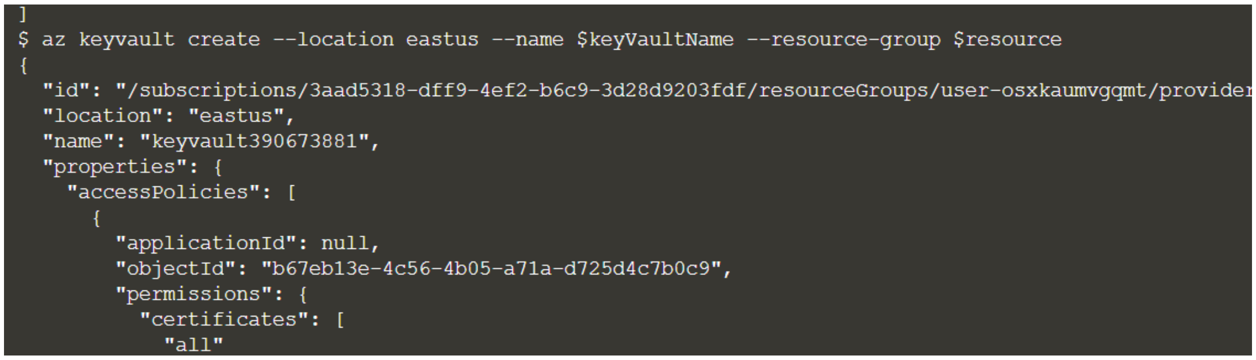
Use the following command to create your first Azure Key Vault instance:

This command might take a few moments!

az keyvault create --location eastus --name $keyVaultName --resource-group $resource

Let's take a look at a few command parameters:

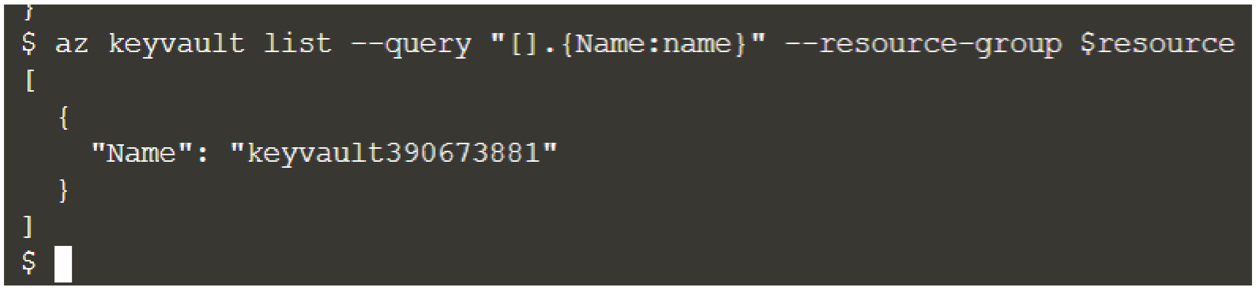
--resource-group: The name of the parent resource group to create the new Azure Key Vault in.
--name: The name for your new Key Vault.
--enable-purge-protection: The purge protection parameter comes hand-in-hand with the soft-delete. If disabled, you can permanently purge (delete) the soft-deleted entities even before the retention period is over. It is recommended that you enable this setting. We are disabling this setting here only because we are deleting our Key Vault at the end of this lab.



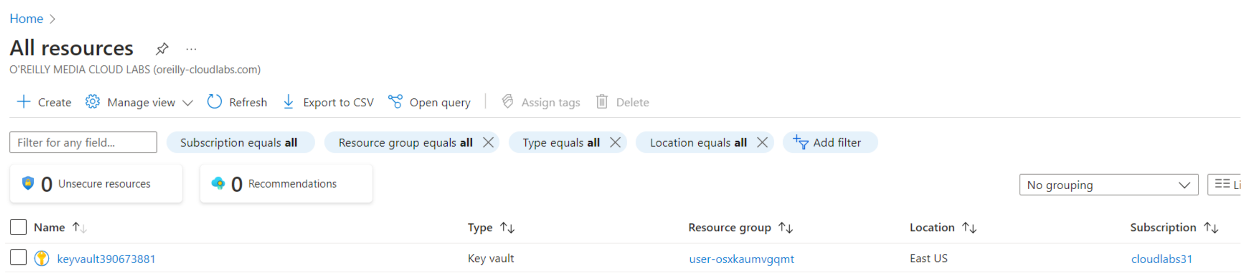
```
]
$ az keyvault create --location eastus --name $keyVaultName --resource-group $resource
{
  "id": "/subscriptions/3aad5318-dff9-4ef2-b6c9-3d28d9203fdf/resourceGroups/user-osxkaumvgqmt/provider
  "location": "eastus",
  "name": "keyvault390673881",
  "properties": {
    "accessPolicies": [
      {
        "applicationId": null,
        "objectId": "b67eb13e-4c56-4b05-a71a-d725d4c7b0c9",
        "permissions": {
          "certificates": [
            "all"
```

az keyvault list --query "[].{Name:name}" --resource-group $resource

Make sure you see the your new Key Vault in the JSON list under the Name property.



```
}
$ az keyvault list --query "[].{Name:name}" --resource-group $resource
[
  {
    "Name": "keyvault390673881"
  }
]
$ 
```


Home >

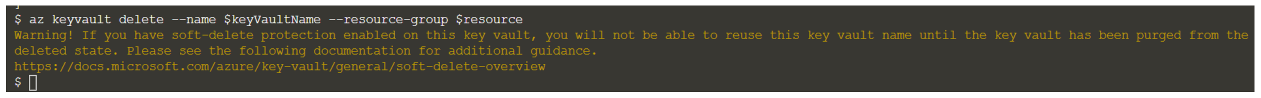
All resources

O'REILLY MEDIA CLOUD LABS (oreilly-cloudlabs.com)

Name	Type	Resource group	Location	Subscription
keyvault390673881	Key vault	user-osxkaumvgqmt	East US	cloudlabs31

Time to purge our Azure Key Vault.

Note: The soft-delete option is enabled for all new Key Vaults. This means deleting a Key Vault should be followed by a purge operation.


```
$ az keyvault delete --name $keyVaultName --resource-group $resource
Warning! If you have soft-delete protection enabled on this key vault, you will not be able to reuse this key vault name until the key vault has been purged from the
deleted state. Please see the following documentation for additional guidance.
https://docs.microsoft.com/azure/key-vault/general/soft-delete-overview
$ 
```

you would have to run the following command to purge (permanently delete) your Key Vault:

az keyvault purge --name $keyVaultName

Working with Azure Key Vaults: Update an Existing Azure Key Vault

Log into the Azure CLI
Update the Azure Key Vault configuration using the CLI
Check the Key Vault configuration in the Azure portal
Clean up Azure Key Vault

Updating Azure Key Vault Configuration Using CLI
There are several Azure Key Vault configuration, which can be changed in the Azure portal, or programmatically using tools such as the Azure CLI. A few of these options are as follows:

Purge protection: When enabled, soft-deleted Key Vaults can not be purged/deleted before the retention period is over. It is recommended that you enable this option for your Production Key Vaults.
Retention period: During this period (in days), a soft-deleted Key Vault, secret, key, or certificate is recoverable. This setting comes hand-in-hand with soft-delete and purge-protection. Don't forget that you still can permanently delete your Key Vault during the retention period by purging it, given that purge-protection is NOT enabled.
Public network access: When enabled, your Key Vault can be used by clients in the public network and Internet. When disabled, only specific clients that are part of an Azure private virtual network can access the Key Vault.

Enabled for disk encryption: Keys stored in Azure Key Vault can be used to encrypt Azure Disks attached to Azure VMs (virtual machines). You need to turn this option ON so this feature can be used.
Enabled for template deployment: Entities stored in Azure Key Vault can be used when deploying resources using ARM templates. You need to turn this option ON so this feature can be used. ARM templates allow you to represent Azure resources as a JSON file so it can be stored in the source control or used in automation scenarios.

Let's use the following command to enable disk encryption for the Key Vault and enable public network access:
This command might take a few moments!

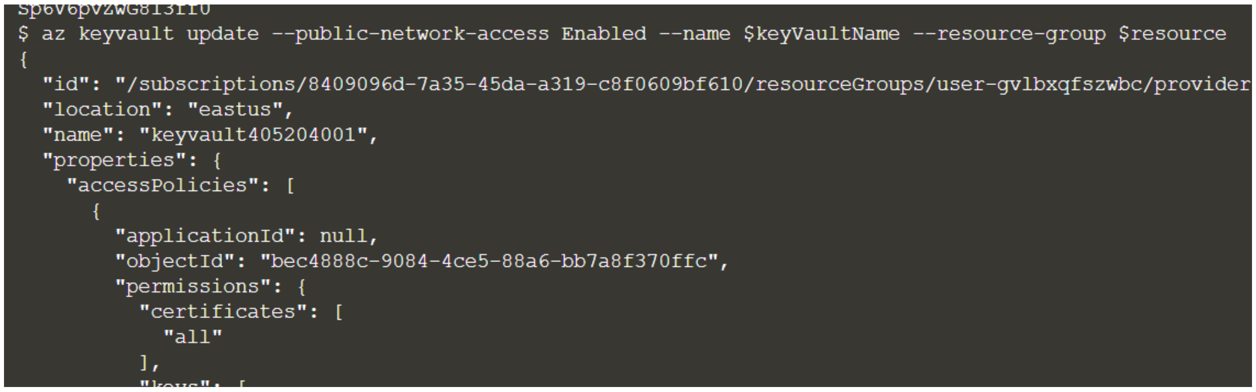
az keyvault update --public-network-access Enabled --name $keyVaultName --resource-group $resource

Let's take a look at the command parameters:

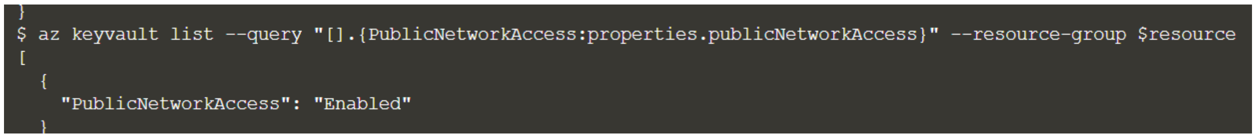
--resource-group: The name of the parent resource group to create the new Azure Key Vault in.
--name: The name for your new Key Vault.
--enabled-for-disk-encryption: When enabled, the Key Vault keys can be used by Azure Disks for disk encryption.
--public-network-access: Allowing clients in public networks (the internet) to access the Key Vault.

You can use the following command to check the Key Vault configuration details:

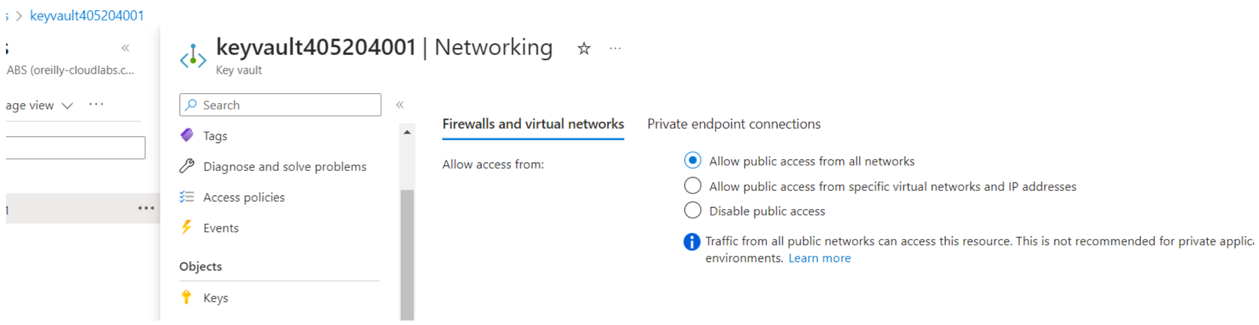
az keyvault list --query "[].{PublicNetworkAccess:properties.publicNetworkAccess}" --resource-group $resource



```
Sp6V6pVZWG813110
$ az keyvault update --public-network-access Enabled --name $keyVaultName --resource-group $resource
{
  "id": "/subscriptions/8409096d-7a35-45da-a319-c8f0609bf610/resourceGroups/user-gvlbxqfszwbc/provider
  "location": "eastus",
  "name": "keyvault405204001",
  "properties": {
    "accessPolicies": [
      {
        "applicationId": null,
        "objectId": "bec4888c-9084-4ce5-88a6-bb7a8f370ffc",
        "permissions": {
          "certificates": [
            "all"
          ],
```



```
}
$ az keyvault list --query "[].{PublicNetworkAccess:properties.publicNetworkAccess}" --resource-group $resource
[
  {
    "PublicNetworkAccess": "Enabled"
  }
```


keyvault405204001 | Networking

Key vault

Search
Tags
Diagnose and solve problems
Access policies
Events
Objects
Keys
Firewalls and virtual networks
Private endpoint connections

Allow access from:

[✓ Allow public access from all networks]
Allow public access from specific virtual networks and IP addresses
Disable public access

Traffic from all public networks can access this resource. This is not recommended for private applic
environments. Learn more

Note: The soft-delete option is enabled for all new Key Vaults. This means deleting a Key Vaults should be followed by a purge operation.

```
az keyvault delete --name $keyVaultName --resource-group $resource
```

Working with Azure Key Vaults: Work with Certificates

Log into the Azure CLI
Working with Azure Key Vault certificates using the Azure CLI
Check the new Key Vault certificate in the Azure portal
Delete, recover, and purge a certificate
Clean up Azure Key Vault

Understanding Azure Key Vault Certificate
Azure Key Vault certificates enable you to store your X.509 certificates in a safe and secure vault. Having the right permissions means that your client services (and other Azure servicves, such as App Services) can access these certificates at runtime and use them.

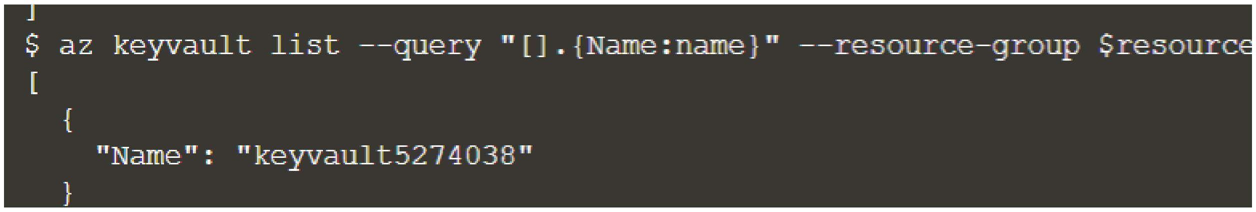
One use case for these certificates is securing your website domain names. Imagine you bought a custom domain for your website, such as johnsmith.com. To use HTTPS (SSL) with this domain (e.g., accessing https://johnsmith.com), you need to purchase an SSL (X.509) certificate for this domain. This certificate should be signed by a certificate authority (CA), and consists of one or more files. These certificates can be stored in Azure Key Vault.

Using Azure Key Vault certificate support, you can:

Create or import a X.509 certificate
Manage different versions of a certificate
Control permissions and access policies for the certificate
Implement automatic certificate renewals

Create a New Azure Key Vault Certificate
Use the following command to confirm your Key Vault is created and ready

```
az keyvault list --query "[].{Name:name}" --resource-group $resource
```



```
$ az keyvault list --query "[].{Name:name}" --resource-group $resource
[
  {
    "Name": "keyvault5274038"
  }
```

Certificates can be imported or generated directly in the Key Vault. Use the following command to generate your first certificate:

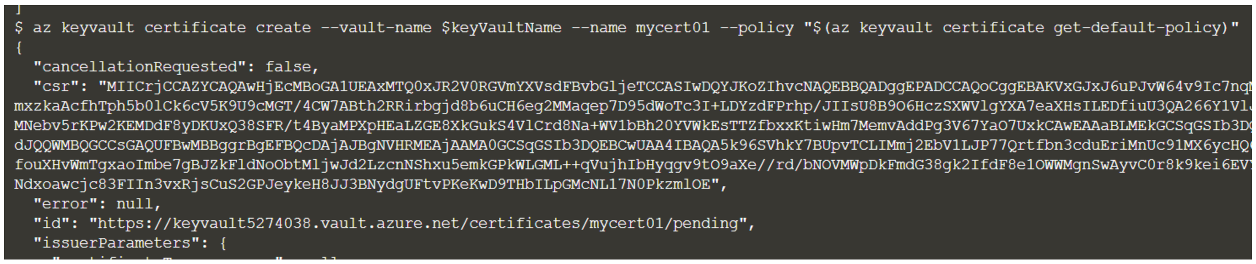
az keyvault certificate create --vault-name $keyVaultName --name mycert01 --policy "$(az keyvault certificate get-default-policy)"

Let's take a look at the command parameters:

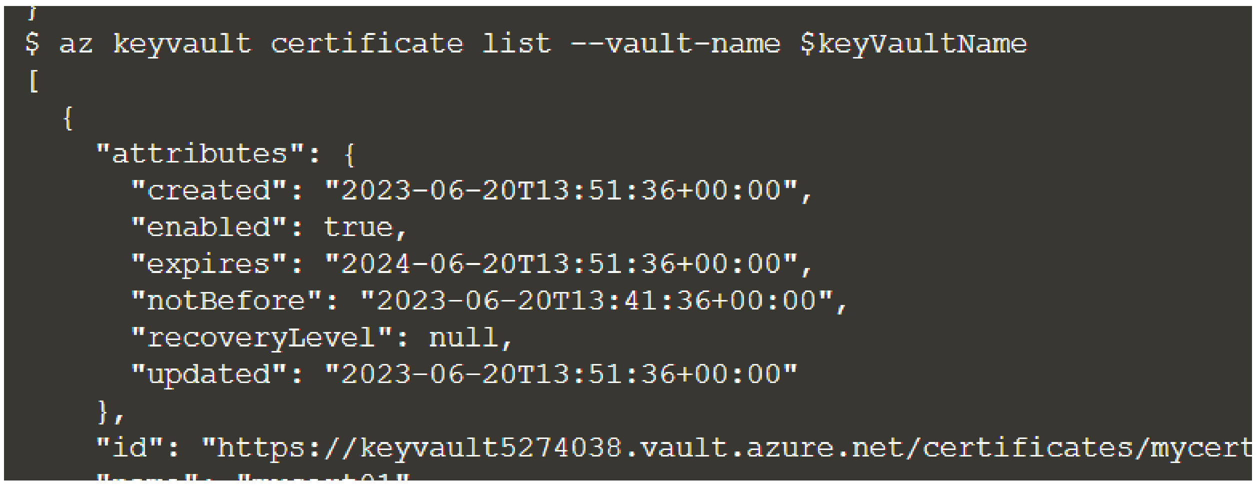
--name: The name for the new Key Vault certificate
--vault-name: The name of your existing Key Vault
--policy: According to Microsoft, "A certificate policy contains information on how to create and manage lifecycle of a Key Vault certificate." In this example, we simply use the default policy.

You can use the following command to list all certificates under a certain Key Vault:

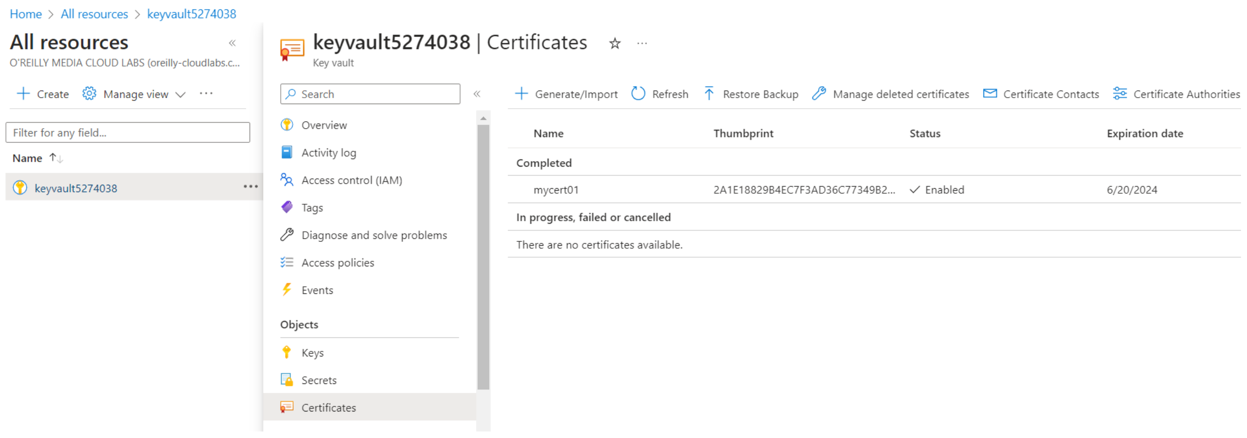
az keyvault certificate list --vault-name $keyVaultName



```
$ az keyvault certificate create --vault-name $keyVaultName --name mycert01 --policy "$(az keyvault certificate get-default-policy)"
{
  "cancellationRequested": false,
  "csr": "MIICrjCCAZYCAQAwHjEcMBoGA1UEAxMTQ0xJR2V0RGVmYXVsdFBvbGljeTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKVxGJxJ6uPJvW64v9Ic7nq
mxzkaAcfhTph5b0lCk6cV5K9U9cMGT/4CW7ABth2RRirbgjd8b6uCH6eg2MMaqep7D95dWoTc3I+LDYzdFPrhp/JIIsU8B9O6HczSXWVlgYXA7eaXHsILEDfiuU3QA266Y1Vl
MNebv5rKPw2KEMDdF8yDKUxQ38SFR/t4ByaMPXpHEaLZGE8XkGukS4VlCrd8Na+WV1bBh20YVWkEsTTZfbxxKtiwHm7MemvAddPg3V67YaO7UxkCAwEAAaBLMEkGCSqGSIb3D
dJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAJBgNVHRMEAjAAMA0GCSqGSIb3DQEBCwUAA4IBAQA5k96SVhkY7BUpvTCLIMmj2EbV1LJP77Qrtfbn3cduEriMnUc91MX6ycHQ
fouXHvWmTgxaoImbe7gBJZkFldNoObtMljwJd2LzcnNShxu5emkGPkWLGML++qVujhIbHyqgv9tO9aXe//rd/bNOVMWpDkFmdG38gk2IfdF8e1OWWMgnSwAyvC0r8k9kei6EV
Ndxoawcjc83FIIn3vxRjsCuS2GPJeykeH8JJ3BNydgUFtvPKeKwD9THbILpGMcNL17N0PkzmlOE",
  "error": null,
  "id": "https://keyvault5274038.vault.azure.net/certificates/mycert01/pending",
  "issuerParameters": {
```



```
$ az keyvault certificate list --vault-name $keyVaultName
[
  {
    "attributes": {
      "created": "2023-06-20T13:51:36+00:00",
      "enabled": true,
      "expires": "2024-06-20T13:51:36+00:00",
      "notBefore": "2023-06-20T13:41:36+00:00",
      "recoveryLevel": null,
      "updated": "2023-06-20T13:51:36+00:00"
    },
    "id": "https://keyvault5274038.vault.azure.net/certificates/mycert
```



Name	Thumbprint	Status	Expiration date
Completed			
mycert01	2A1E18829B4EC7F3AD36C77349B2...	✓ Enabled	6/20/2024
In progress, failed or cancelled			
There are no certificates available.			

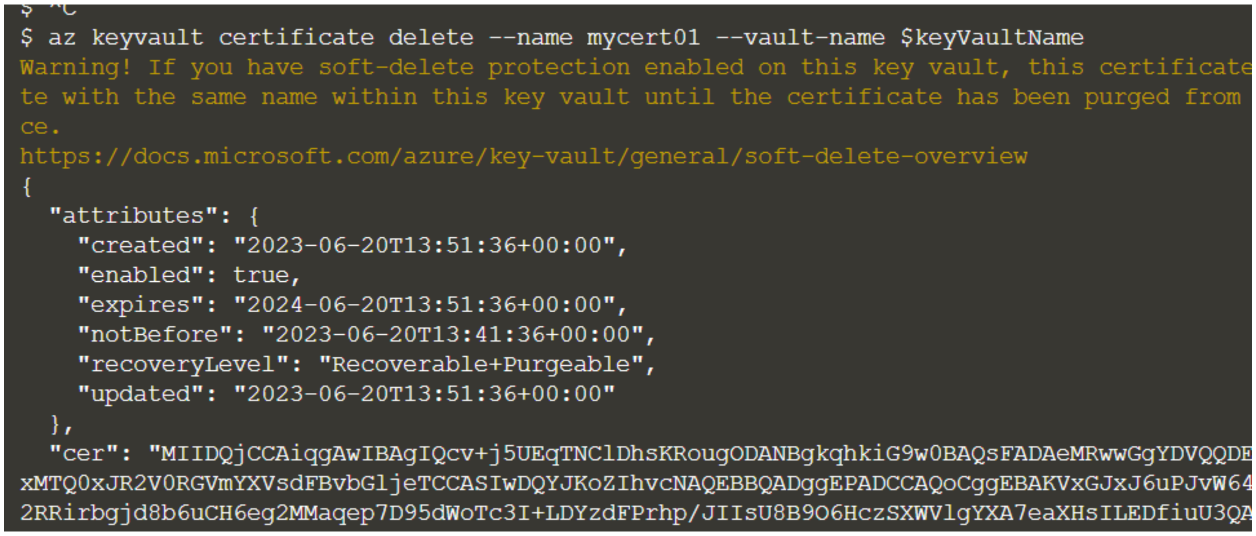
Soft-delete is enabled by default for all the Key Vaults. Let's see how a certificate can be soft-deleted, recovered, and purged in the next step.

Before cleaning up the Key Vault, let's work with certificate deletion and recovery.

As mentioned, soft-delete is enabled by default for all the Key Vaults and can't be turned of. This means we should be able to delete a certificate and recover it.

First use the following command to delete your certificate:

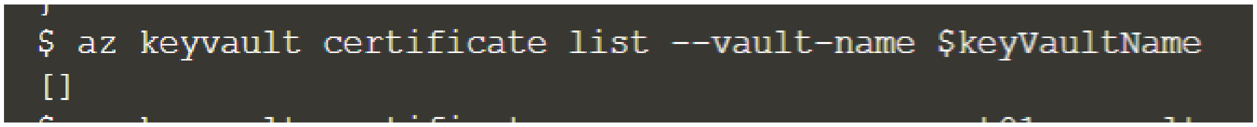
az keyvault certificate delete --name mycert01 --vault-name $keyVaultName



```
$ ^C
$ az keyvault certificate delete --name mycert01 --vault-name $keyVaultName
Warning! If you have soft-delete protection enabled on this key vault, this certificate
te with the same name within this key vault until the certificate has been purged from
ce.
https://docs.microsoft.com/azure/key-vault/general/soft-delete-overview
{
  "attributes": {
    "created": "2023-06-20T13:51:36+00:00",
    "enabled": true,
    "expires": "2024-06-20T13:51:36+00:00",
    "notBefore": "2023-06-20T13:41:36+00:00",
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2023-06-20T13:51:36+00:00"
  },
  "cer": "MIIDQjCCAiqgAwIBAgIQcv+j5UEqTNClDhsKRougODANBgkqhkiG9w0BAQsFADAeMRwwGgYDVQQDE
xMTQ0xJR2V0RGVmYXVsdFBvbGljeTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKVxGJxJ6uPJvW64
2RRirbgjd8b6uCH6eg2MMaqep7D95dWoTc3I+LDYzdFPrhp/JIIsU8B9O6HczSXWVlgYXA7eaXHsILEDfiuU3QA
```

This command deletes our certificate, but fortunately we have soft-delete enabled. Use the following command to prove that the certificate is indeed deleted and not visible anymore:

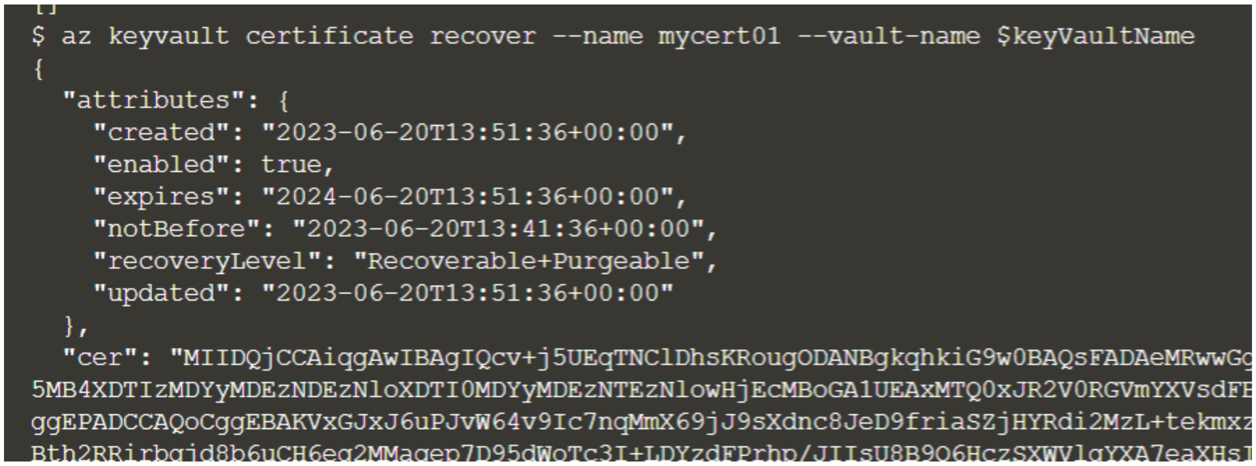
az keyvault certificate list --vault-name $keyVaultName



```
}
$ az keyvault certificate list --vault-name $keyVaultName
[]
```

Now, time to recover the certificate using this command:

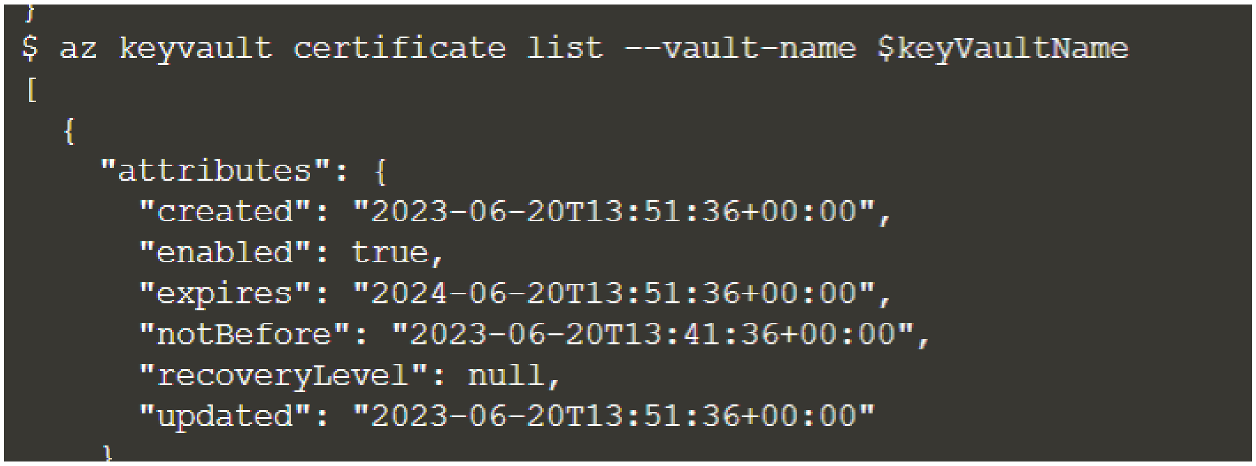
az keyvault certificate recover --name mycert01 --vault-name $keyVaultName



```
[]
$ az keyvault certificate recover --name mycert01 --vault-name $keyVaultName
{
  "attributes": {
    "created": "2023-06-20T13:51:36+00:00",
    "enabled": true,
    "expires": "2024-06-20T13:51:36+00:00",
    "notBefore": "2023-06-20T13:41:36+00:00",
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2023-06-20T13:51:36+00:00"
  },
  "cer": "MIIDQjCCAiqgAwIBAgIQcv+j5UEqTNClDhsKRougODANBgkqhkiG9w0BAQsFADAeMRwwGg
5MB4XDTIzMDYyMDEzNDEzNloXDTI0MDYyMDEzNTEzNlowHjEcMBoGA1UEAxMTQ0xJR2V0RGVmYXVsdFB
ggEPADCCAQoCggEBAKVxGJxJ6uPJvW64v9Ic7nqMmX69jJ9sXdnc8JeD9friaSZjHYRdi2MzL+tekmxz
Bth2RRirbqid8b6uCH6eq2MMagep7D95dWoTc3I+LDYzdFPrhp/JIIsU8B9O6HczSXWVlqYXA7eaXHsI
```

Run the following command again:

az keyvault certificate list --vault-name $keyVaultName



```
}
$ az keyvault certificate list --vault-name $keyVaultName
[
  {
    "attributes": {
      "created": "2023-06-20T13:51:36+00:00",
      "enabled": true,
      "expires": "2024-06-20T13:51:36+00:00",
      "notBefore": "2023-06-20T13:41:36+00:00",
      "recoveryLevel": null,
      "updated": "2023-06-20T13:51:36+00:00"
    }
```

Let's delete the certificate again:

az keyvault certificate delete --name mycert01 --vault-name $keyVaultName you had to also run the following purge command to permanently purge the certificate, key, or secret:

```
az keyvault certificate purge --name mycert01 --vault-name $keyVaultName
```

Use the following command to delete/soft-delete the Key Vault:

```
az keyvault delete --name $keyVaultName --resource-group $resource
```

the following command to purge (permanently delete) your Key Vault:

```
az keyvault purge --name $keyVaultName
```

Working with Azure Key Vaults: Work with Keys
Log into the Azure CLI
Work with Azure Key Vault keys using the Azure CLI
Check the new Key Vault key in the Azure portal
Delete, recover, and purge a key
Clean up Azure Key Vault

Understanding Azure Key Vault Keys
Azure Key Vault keys enable you to store encryption keys (a.k.a. cryptographic key) in a safe and secure vault. With the right permissions, your client services (and other Azure services, such as App Services, Azure Disks, Cosmos DB, etc.) can use these key to encrypt data in transit and data at rest.

One use case for these keys is encrypting Azure Virtual Machine Disks. Imagine you provisioned an Azure VM and would like to control the encryption key used to encrypt this VM's disk. You simply import or create a new key in Azure Key Vault and configure your Azure VM to read the encryption key from your Azure Key Vault.

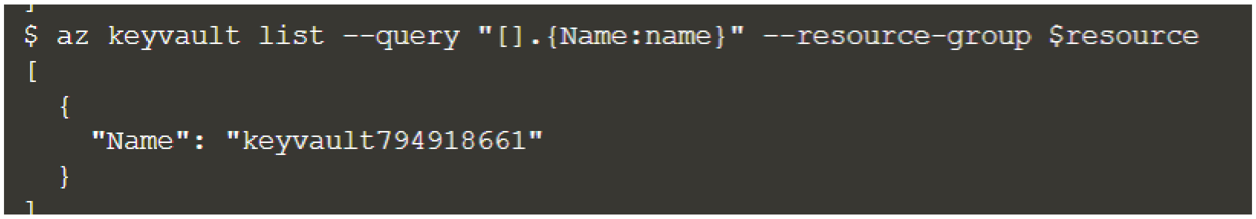
Several Azure services can use with Azure Key Vault keys. A few include Azure Disks, Azure Cosmos DB, Azure SQL Database, Azure Storage Account, and more.

We have preconfigured this lab so a new Azure Key Vault is created for you. In the next step, we will add a key to this Key Vault.

Create a New Azure Key Vault Key
Now that we've introduced Azure Key Vault keys, let's create one.

Use the following command to confirm your Key Vault is created and ready:

```
az keyvault list --query "[].{Name:name}" --resource-group $resource
```



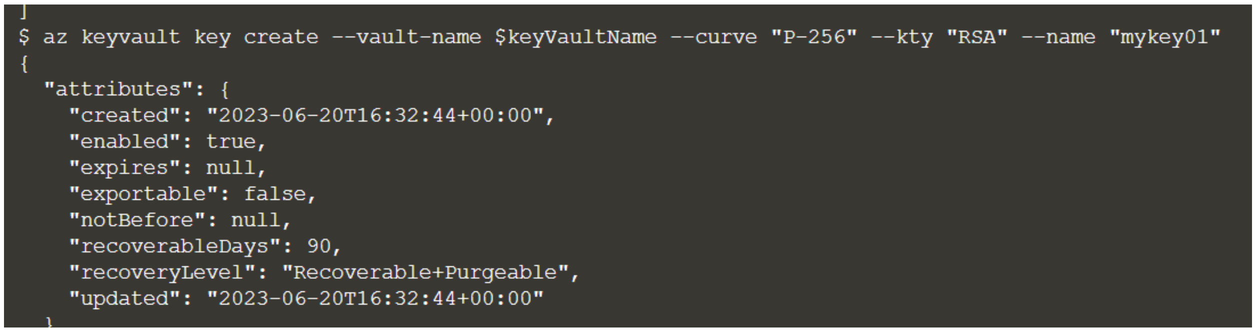
```
$ az keyvault list --query "[].{Name:name}" --resource-group $resource
[
  {
    "Name": "keyvault794918661"
  }
```

Keys can be imported or generated directly in the Key Vault. Use the following command to generate your first key:

az keyvault key create --vault-name $keyVaultName --curve "P-256" --kty "RSA" --name "mykey01"

A key has a few properties which need to be passed to this command as parameters. Getting into cryptographic key details and concepts is beyond the scope of this lab. Take a look at the below resources for more information:

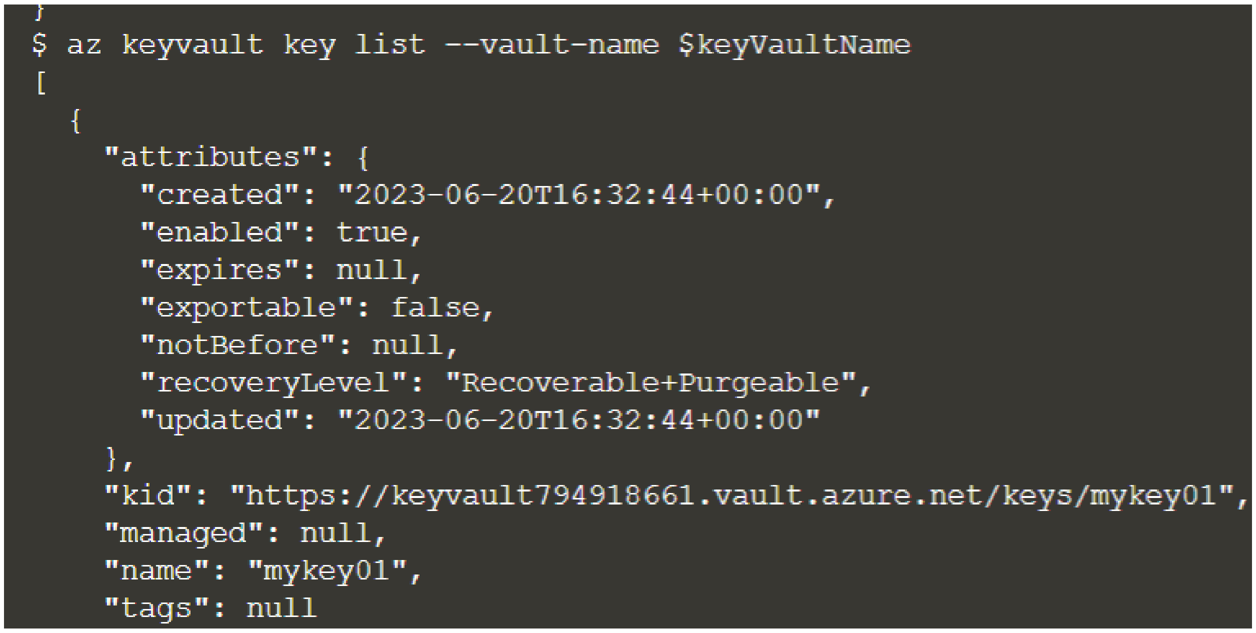
--name: The name for the new Key Vault key.
--vault-name: Name of your existing Key Vault.
--curve: Elliptic curve name. Accepted values are P-256, P-256K, P-384, and P-521.
--kty: The type of the new key. Accepted values are EC, EC-HSM, RSA, RSA-HSM, oct, ct-HSM. Note that HSM stands for Hardware Security Module. Keys with the -HSM suffix are protected by secure hardware modules, the rest are protected by software.
–size: The key size in bits. For example: 2048, 3072, or 4096 for RSA. 128, 192, or 256 for oct. The bigger the size, the more secure the key will be.



```
$ az keyvault key create --vault-name $keyVaultName --curve "P-256" --kty "RSA" --name "mykey01"
{
  "attributes": {
    "created": "2023-06-20T16:32:44+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2023-06-20T16:32:44+00:00"
```

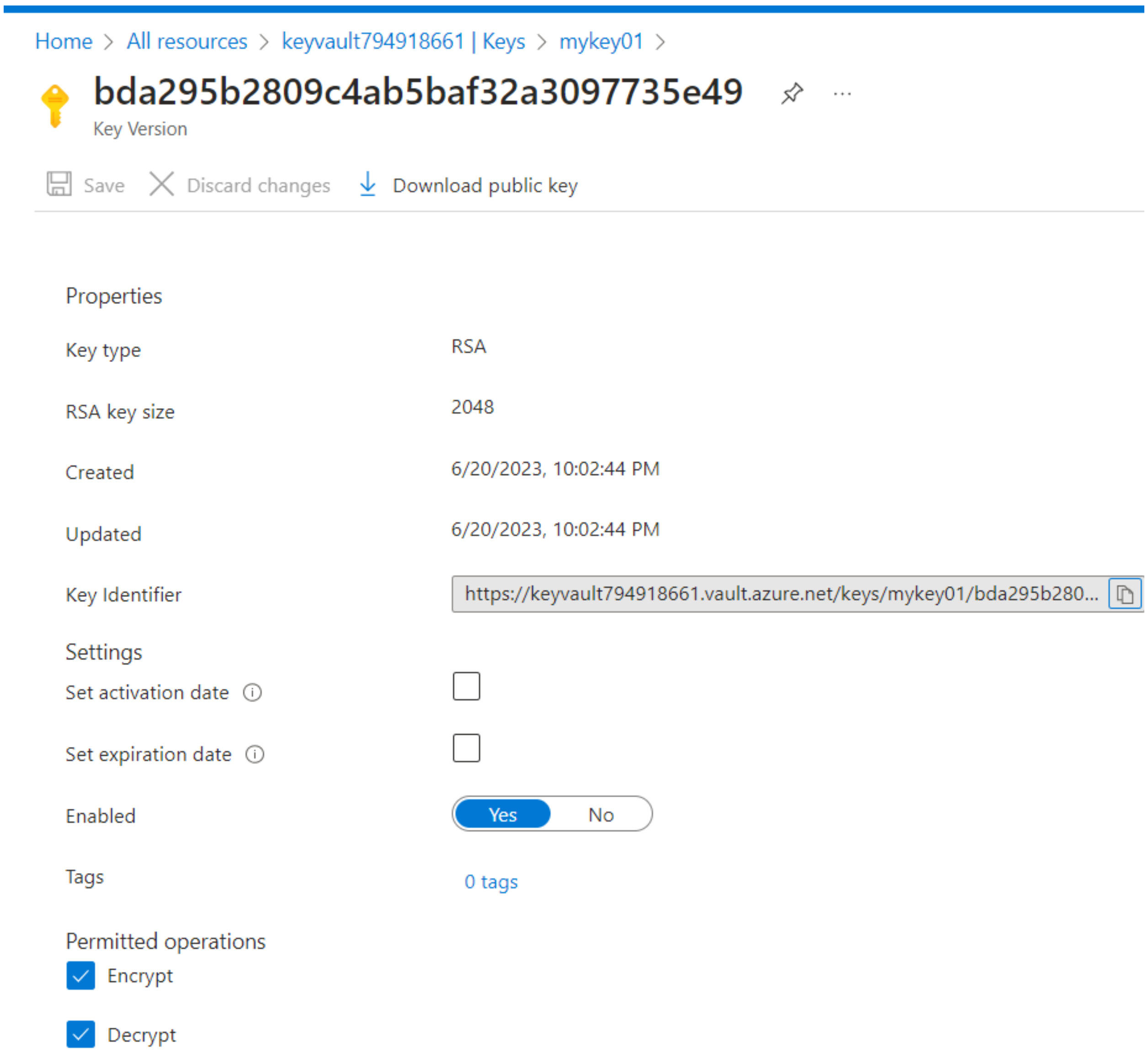
You can use the following command to list all keys under a certain Key Vault:

az keyvault key list --vault-name $keyVaultName



```
}
$ az keyvault key list --vault-name $keyVaultName
[
  {
    "attributes": {
      "created": "2023-06-20T16:32:44+00:00",
      "enabled": true,
      "expires": null,
      "exportable": false,
      "notBefore": null,
      "recoveryLevel": "Recoverable+Purgeable",
      "updated": "2023-06-20T16:32:44+00:00"
    },
    "kid": "https://keyvault794918661.vault.azure.net/keys/mykey01",
    "managed": null,
    "name": "mykey01",
    "tags": null
```

In the next step, we'll check our new key in the Azure portal.
Soft-delete is enabled by default for all the Key Vaults. Let's see how a key can be soft-deleted, recovered, and purged in the next step.


Home > All resources > keyvault794918661 | Keys > mykey01 >

bda295b2809c4ab5baf32a3097735e49

Key Version

Save | Discard changes | Download public key

Properties

Key type	RSA
RSA key size	2048
Created	6/20/2023, 10:02:44 PM
Updated	6/20/2023, 10:02:44 PM
Key Identifier	https://keyvault794918661.vault.azure.net/keys/mykey01/bda295b280...

Settings

Set activation date	☐
Set expiration date	☐
Enabled	[✓ Yes] No
Tags	0 tags

Permitted operations

✓ Encrypt

✓ Decrypt

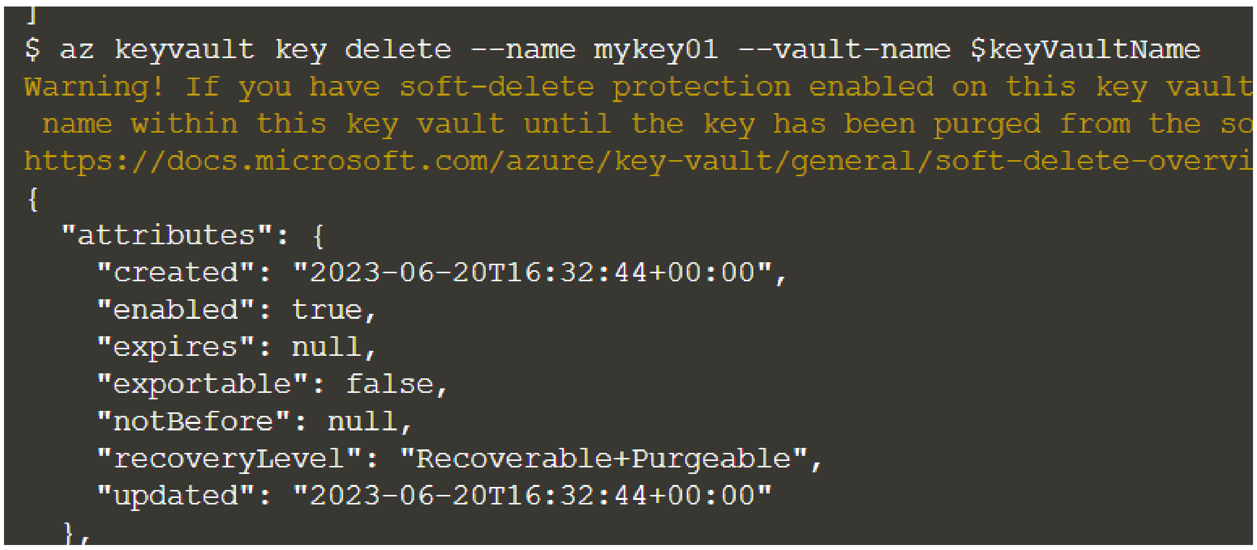
Delete, Recover, and Purge a Key
Before cleaning up the Key Vault, let's work with key deletion and recovery in this step.

As mentioned, soft-delete is enabled by default for all the Key Vaults, and can't be turned of. This means we should be able to delete a key and recover it.

First use the following command to delete your key:

```
az keyvault key delete --name mykey01 --vault-name $keyVaultName
```

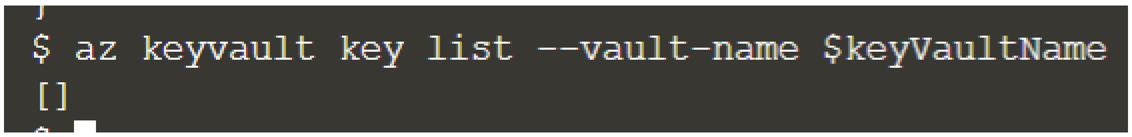
Wait for the command to succeed.



```
]
$ az keyvault key delete --name mykey01 --vault-name $keyVaultName
Warning! If you have soft-delete protection enabled on this key vault
 name within this key vault until the key has been purged from the so
https://docs.microsoft.com/azure/key-vault/general/soft-delete-overvi
{
  "attributes": {
    "created": "2023-06-20T16:32:44+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "notBefore": null,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2023-06-20T16:32:44+00:00"
  },
```

This command deletes our key, but fortunately we have soft-delete enabled. Use the following command to prove that the key is indeed deleted, and not visible anymore:

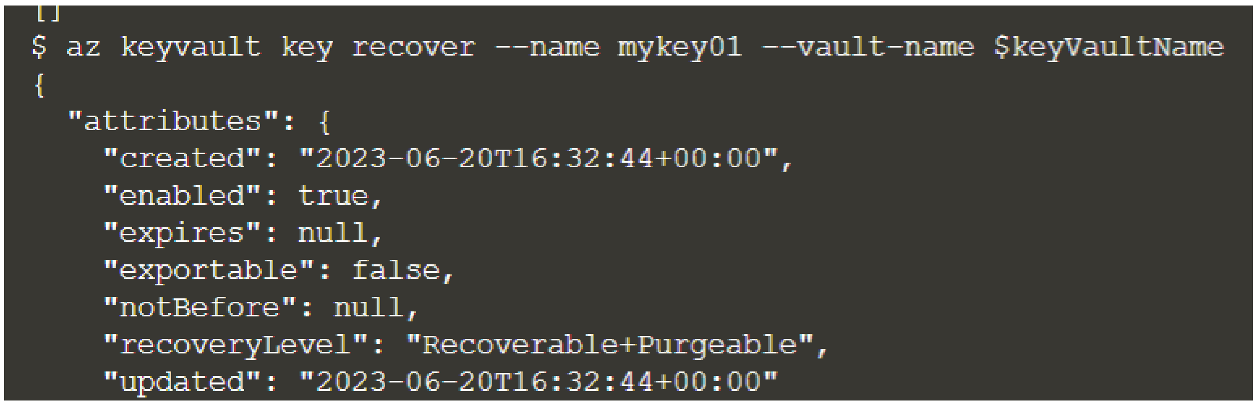
az keyvault key list --vault-name $keyVaultName



```
]
$ az keyvault key list --vault-name $keyVaultName
[]
```

Now, time to recover the key using this command:

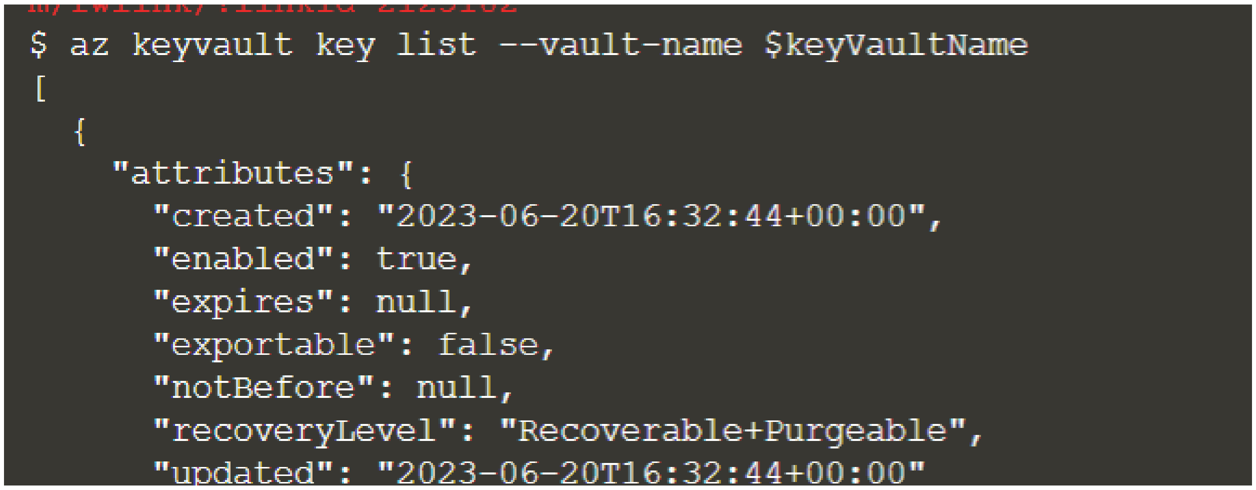
az keyvault key recover --name mykey01 --vault-name $keyVaultName



```
[]
$ az keyvault key recover --name mykey01 --vault-name $keyVaultName
{
  "attributes": {
    "created": "2023-06-20T16:32:44+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "notBefore": null,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2023-06-20T16:32:44+00:00"
```

Run the following command again:

az keyvault key list --vault-name $keyVaultName



```
$ az keyvault key list --vault-name $keyVaultName
[
  {
    "attributes": {
      "created": "2023-06-20T16:32:44+00:00",
      "enabled": true,
      "expires": null,
      "exportable": false,
      "notBefore": null,
      "recoveryLevel": "Recoverable+Purgeable",
      "updated": "2023-06-20T16:32:44+00:00"
```

az keyvault key purge --name mykey01 --vault-name $keyVaultName

In the next step, we will clean up the Key Vault.

Important: Purging a parent Key Vault will also permanently purge all the certificates, keys, and secrets in that vault!

Note: The soft-delete option is enabled for all new Key Vaults. This means deleting a Key Vaults should be followed by a purge operation.

Use the following command to delete/soft-delete the Key Vault:

az keyvault delete --name $keyVaultName --resource-group $resource

to purge (permanently delete) your Key Vault:

az keyvault purge --name $keyVaultName

Working with Azure Key Vaults: Work with Secrets

Log into the Azure CLI
Work with Azure Key Vault secrets using the Azure CLI
Check the new Key Vault secret in the Azure portal
Delete, recover, and purge a secret
Clean up Azure Key Vault

Understanding Azure Key Vault Secrets
Azure Key Vault secrets enable you to store any custom secret needed by your applications. A secret could be a database connection string, a password, or even a keyword. The common property of all secrets is their type, which is always string.

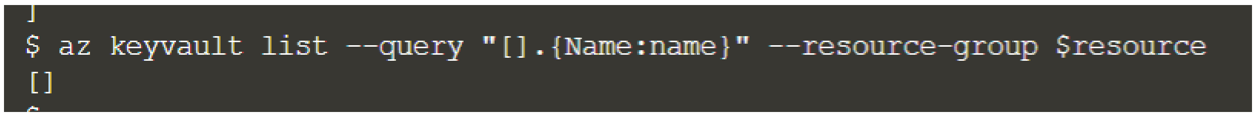
One use case for secrets is storing a database connection string. Imagine you have an App Service that needs to talk to a Cosmos DB database. Instead of hard-coding the database connection string in the code or storing it in the App Service configuration settings, you should store the connection string as a secret in a Key Vault and have your application read it at runtime when needed. This setup will dramatically improve the security of your solution. Your database connection string would be safe, even if the code or configuration file was compromised.

Several Azure services have native support for Key Vault secrets. A few include Azure App Services and Azure Function Apps. Both these services can access Azure Key Vault secrets using a method called Azure Key Vault references. Using this method, you can use Key Vault secrets without changing your app code.

Create a New Azure Key Vault Secret
Now that we've introduced Azure Key Vault secrets, let's create one.

Use the following command to confirm your Key Vault is created, and ready:

az keyvault list --query "[].{Name:name}" --resource-group $resource



```
$ az keyvault list --query "[].{Name:name}" --resource-group $resource
[]
```

Use the following command to generate your first key:

az keyvault secret set --name mySQLPassword --value "WeakPassword" --vault-name $keyVaultName

Let's take a look at the command parameters:

--name: The name for the new Key Vault secret
--vault-name: Name of your existing Key Vault
--value: The value for our secret

You can use the following command to list all keys under a certain Key Vault:

az keyvault secret list --vault-name $keyVaultName

As mentioned, soft-delete is enabled by default for all the Key Vaults and can't be turned of. This means we should be able to delete a secret and recover it.

First use the following command to delete your secret:

```
az keyvault secret delete --name mySQLPassword --vault-name $keyVaultName
```

This command deletes our secret, but fortunately we have soft-delete enabled. Use the following command to prove that the key is indeed deleted, and not visible anymore:

```
az keyvault secret list --vault-name $keyVaultName
```

Now, time to recover the secret using this command:

```
az keyvault secret recover --name mySQLPassword --vault-name $keyVaultName
```

Run the following command again:

```
az keyvault secret list --vault-name $keyVaultName
```

Let's delete the secret again:

```
az keyvault secret delete --name mySQLPassword --vault-name $keyVaultName
```

purge command to permanently purge the certificate, key, or secret:

```
az keyvault secret purge --name mySQLPassword --vault-name $keyVaultName
```

Important: Purging a parent Key Vault will also permanently purge all the certificates, keys, and secrets in that vault!Note: The soft-delete option is enabled for all new Key Vaults. This means deleting a Key Vaults should be followed by a purge operation.

Use the following command to delete/soft-delete the Key Vault:

```
az keyvault delete --name $keyVaultName --resource-group $resource
```

purge (permanently delete) your Key Vault:

```
az keyvault purge --name $keyVaultName
```

Working with Azure Key Vaults: Delete, Recover, and Purge an Azure Key Vault

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

Log into the Azure CLI
Soft-delete an Azure Key Vault using the Azure CLI
Recover a soft-deleted Azure Key Vault using the Azure CLI

Purge Azure Key Vault using the Azure CLI

In the previous labs, we soft-deleted certificates, keys, and secrets. In this lab, we will soft-delete the parent Key Vault itself.

Use the following command to soft-delete our Key Vault:

```
az keyvault delete --name $keyVaultName --resource-group $resource
```

Use the following command to confirm the Key Vault is deleted:

```
az keyvault list --resource-group $resource
```

Let's see the recover command in the next step.

the following command to recover the soft-deleted Key Vault:

```
az keyvault recover --location eastus --name $keyVaultName --resource-group $resource
```

Purge an Azure Key Vault
Time to purge our Azure Key Vault.

Note: The soft-delete option is enabled for all new Key Vaults. This means deleting a Key Vaults should be followed by a purge operation.

Use the following command to delete/soft-delete the Key Vault:

```
az keyvault delete --name $keyVaultName --resource-group $resource
```

command to purge (permanently delete) your Key Vault:

```
az keyvault purge --name $keyVaultName
```

In this lab, you will learn to work with Azure Key Vault access policies using the Azure CLI.

You can control access to Azure Key Vault keys, secrets, and certificates using one of the following access models:

Role-based Access Control (RBAC)
Key Vault access policies

To log in to your Azure subscription: `az login -u $username -p $password`

To create a new Azure Key Vault service: az keyvault create --location eastus --name $keyVaultName --resource-group $resource

To create a new Azure VM with a system-assigned identity: az vm create --location eastus --name $vmName --size Standard_DS2_v2 --resource-group $resource --public-ip-sku Standard --assign-identity [system] --image UbuntuLTS --admin-username katauser --generate-ssh-keys

Understand Azure Key Vault Access Policies
One of the methods in which you can control access to Azure Key Vault keys, secrets, and certificates is by using the Azure Key Vault access policies.

Imagine you have an application running on an Azure virtual machine, and this application needs permission to read some secrets from an Azure Key Vault. In order to grant this access, you can create a Key Vault access policy that grants the Get Secret permission to the Azure VM.

As per Microsoft, "A Key Vault access policy determines whether a given security principal, namely a user, application or user group, can perform different operations on Key Vault secrets, keys, and certificates."

Create a New Azure Key Vault Access Policy
First, we need to grab the identity ID for the virtual machine and store it in a variable:

VM_OBJECT_ID=$(az vm show --name $vmName --resource-group $resource --query identity.principalId --output tsv)

You can use this command to see the value of the Azure VM ID:

echo $VM_OBJECT_ID

Now, we can create the access policy using this command:

az keyvault set-policy --name $keyVaultName --resource-group $resource --secret-permissions get --object-id $VM_OBJECT_ID

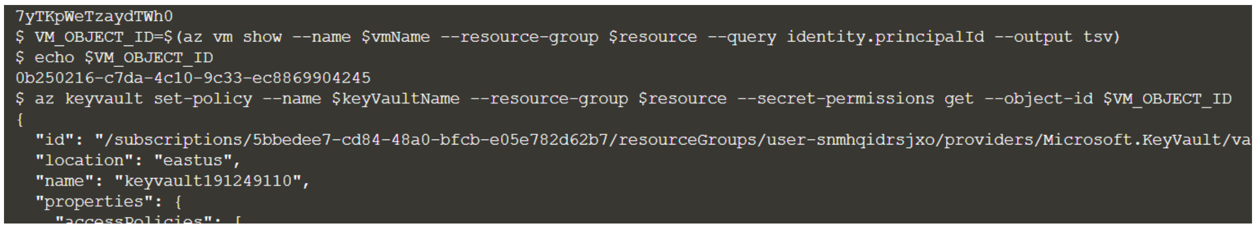
Here are the command parameters:

--name: The Key Vault name
--resource-group: The Key Vault resource group name
--secret-permissions: Space-delimited list of secret permissions you are willing to grant (accepted values are all, backup, delete, get, list, purge, recover, restore, and set)
--object-id: A GUID that identifies the principal that will receive these permissions, in our case the managed identity ID of the Azure VM

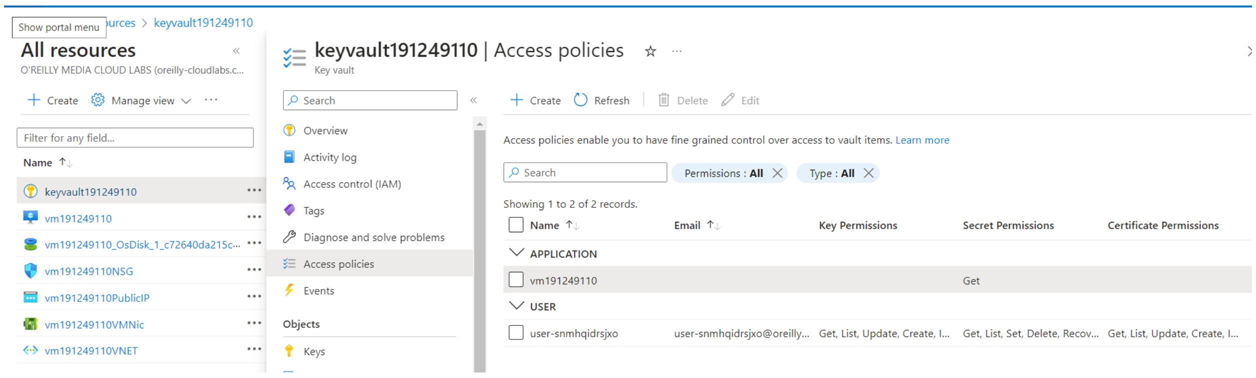
Note: Using the same command, you can grant --key-permissions and --certificate-permissions as well. See the command page for details.

From this point, all applications running on this particular VM can read secrets from your Azure Key Vault instance.

In the next step, we will confirm that the access policy is created.



```
7yTKpWeTzaydTWh0
$ VM_OBJECT_ID=$(az vm show --name $vmName --resource-group $resource --query identity.principalId --output tsv)
$ echo $VM_OBJECT_ID
0b250216-c7da-4c10-9c33-ec8869904245
$ az keyvault set-policy --name $keyVaultName --resource-group $resource --secret-permissions get --object-id $VM_OBJECT_ID
{
  "id": "/subscriptions/5bbedee7-cd84-48a0-bfcb-e05e782d62b7/resourceGroups/user-snmhqidrsjxo/providers/Microsoft.KeyVault/va
  "location": "eastus",
  "name": "keyvault191249110",
  "properties": {
    "accessPolicies": [
```



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

Showing 1 to 2 of 2 records.

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions
APPLICATION				
vm191249110			Get	
USER				
user-snmhqidrsjxo	user-snmhqidrsjxo@oreilly...	Get, List, Update, Create, I...	Get, List, Set, Delete, Recov...	Get, List, Update, Create, I...

Remove Key Vault Access Policies
Now, let's delete the access policy we created using the following command:

az keyvault delete-policy --name $keyVaultName --resource-group $resource --object-id $VM_OBJECT_ID

Here are the command parameters:

--name: The Key Vault name
--resource-group: The Key Vault resource group name
--object-id: A GUID that identifies the principal whose permissions will be revoked, in our case the managed identity ID of the Azure VM
From now on, all applications running on your VM cannot read secrets from the Azure Key Vault.