



Introduction

3 minutes

PetDash is an online pet food delivery company that provides store-to-door service for all their customer's pet needs. They take online orders, store credit cards and personal details in their SQL database, and have a secure website running on Azure App Service to interact with customers. They've been in business a little over a year and Steve, one of the website admins, noticed that their website certificate for the **petdash.com** domain has expired. Steve quickly renews the certificate and gets it installed on the server, and begins to explore ways to ensure that this problem never happens again. The investigation reveals that Azure Key Vault supports certificate management. Even better, Key Vault can communicate with App Service to provide the certification *and* renew it automatically if necessary.

Azure Key Vault helps safeguard cryptographic keys and secrets that cloud applications and services use. Key Vault streamlines the key management process and enables you to maintain control of keys that access and encrypt your data. Developers can create keys for development and testing in minutes, and then migrate them to production keys. Security administrators can grant (and revoke) permission to keys, as needed.

Learning objectives

In this module, you will:

- Explore proper usage of Azure Key Vault
 - Manage access to an Azure Key Vault
 - Explore certificate management with Azure Key Vault
 - Configure a Hardware Security Module Key-generation solution
-

Next unit: Guidelines for using Azure Key Vault

[Continue >](#)



Guidelines for using Azure Key Vault

5 minutes

Azure Key Vault is a centralized cloud service for storing application secrets such as encryption keys, certificates, and server-side tokens. Key Vault helps you control your applications' secrets by keeping them in a single central location and providing secure access, permissions control, and access logging.

There are three primary concepts used in an Azure Key Vault: *vaults*, *keys*, and *secrets*.

Vaults

You use Azure Key Vault to create multiple secure containers, called vaults. Vaults help reduce the chances of accidental loss of security information by centralizing application secrets storage. Organizations will have several key vaults. Each key vault is a collection of cryptographic keys and cryptographically protected data (call them "secrets") managed by one or more responsible individuals within your organization. These key vaults represent the logical groups of keys and secrets for your organization; those that you want to manage together. They are like folders in the file system. Key vaults also control and log the access to anything stored in them.

You can create and manage vaults using command line tools such as Azure PowerShell or the Azure CLI, using the REST API, or through the Azure portal.


For example, here's a sample Azure CLI command line to create a new vault in a resource group:

Azure CLI

 Copy

```
az keyvault create \  
  --resource-group <resource-group> \  
  --name <your-unique-vault-name>
```

Here's the same command using Azure PowerShell:

PowerShell	 Copy
<pre>New-AzKeyVault -Name <your-unique-vault-name> -ResourceGroupName <resource-group></pre>	

Keys

Keys are the central actor in the Azure Key Vault service. A given key in a key vault is a cryptographic asset destined for a particular use such as the asymmetric master key of Microsoft Azure RMS, or the asymmetric keys used for SQL Server TDE (Transparent Data Encryption), CLE (Column Level Encryption) and Encrypted backup.

Microsoft and your apps don't have access to the stored keys directly once a key is created or added to a key vault. Applications must use your keys by calling cryptography methods on the Key Vault service. The Key Vault service performs the requested operation within its hardened boundary. The application never has direct access to the keys.

Keys can be single instanced (only one key exists) or be versioned. In the versioned case, a key is an object with a primary (active) key and a collection of one or more secondary (archived) keys created when keys are rolled (renewed). Key Vault supports asymmetric keys (RSA 2048). Your applications may use these for encryption or digital signatures.

There are two variations on keys in Key Vault: hardware-protected, and software-protected.

Hardware protected keys

The Key Vault service supports using HSMs that provide a hardened, tamper-resistant environment for cryptographic processing and key generation. Azure has dedicated HSMs validated to FIPS 140-2 Level 2 that Key Vault uses to generate or store keys. These HSM-backed keys are always locked to the boundary of the HSM. When you ask the Key Vault service to decrypt or sign with a key, the operation is performed inside an HSM.

You can import keys from your own hardware security modules (HSMs) and transfer them to Key Vault without leaving the HSM boundary. This scenario is often referred to as *bring your own key*, or BYOK. More details on generating your own HSM-protected key and then transferring it to Azure Key Vault is available in the summary of this module. You can also use these Azure HSMs directly through the Microsoft Azure Dedicated Hardware Security Module (HSM) service if you need to migrate HSM-protected apps or maintain a high security compliance requirement.

Software protected keys

Key Vault can also generate and protect keys using software-based RSA and ECC algorithms. In general, software-protected keys offer most of the features as HSM-protected keys except the FIPS 140-2 Level 2 assurance:

- Your key is still isolated from the application (and Microsoft) in a container that you manage
- It's stored *at rest* encrypted with HSMs
- You can monitor usage using Key Vault logs

The primary difference (besides price) with a software-protected key is when cryptographic operations are performed, they are done in software using Azure compute services while for HSM-protected keys the cryptographic operations are performed within the HSM.

Tip

For production use, it's recommended to use HSM-protected keys and use software-protected keys in only test/pilot scenarios. There is an additional charge for HSM-backed keys per-month if the key is used in that month. The summary page has a link to the pricing details for Azure Key Vault.

You determine the key generation type when you create the key. For example, the Azure PowerShell command `Add-AzureKeyVaultKey` has a `Destination` parameter that can be set to either `Software` or `HSM`:

PowerShell

 Copy

```
$key = Add-AzureKeyVaultKey -VaultName 'contoso' -Name 'MyFirstKey' -Destination 'HSM'
```

Secrets

Secrets are small (less than 10K) data blobs protected by a HSM-generated key created with the Key Vault. Secrets exist to simplify the process of persisting sensitive settings that almost every application has: storage account keys, .PFX files, SQL connection strings, data encryption keys, etc.

Key vault uses

With these three elements, an Azure Key Vault helps address the following issues:

- **Secrets management.** Azure Key Vault can securely store (with HSMs) and tightly control access to tokens, passwords, certificates, API keys, and other secrets.
- **Key management.** Azure Key Vault is a cloud-based key management solution, making it easier to create and control the encryption keys used to encrypt your data. Azure services such as App Service integrate directly with Azure Key Vault and can decrypt secrets without knowledge of the encryption keys.
- **Certificate management.** Azure Key Vault is also a service that lets you easily provision, manage, and deploy public and private SSL/TLS certificates for use with Azure and your internal connected resources. It can also request and renew TLS certificates through partnerships with certificate authorities, providing a robust solution for certificate lifecycle management.

Important

Key Vault is designed to store configuration secrets for server applications. It's not intended for storing data belonging to your app's users, and it shouldn't be used in the client-side part of an app. This is reflected in its performance characteristics, API, and cost model.

User data should be stored elsewhere, such as in an Azure SQL database with Transparent Data Encryption, or a storage account with Storage Service Encryption.

Secrets used by your application to access those data stores can be kept in Key Vault.

Best practices

Here are some security best practices for using Azure Key Vault.

Best practice	Solution
Grant access to users, groups, and applications at a specific scope.	Use RBAC's predefined roles. For example, to grant access to a user to manage key vaults, you would assign the predefined role Key Vault Contributor to this user at a specific scope. The scope, in this case, would be a subscription, a resource group, or just a specific key vault. If the predefined roles don't fit your needs, you can define your own roles.
Control what users have access to.	Access to a key vault is controlled through two separate interfaces: management plane, and data plane. The management plane and data plane access controls work independently. Use RBAC to control what users have access to. For example, if you want to grant an application the rights to use keys in a key vault, you only need to grant data plane access permissions by using key vault access policies, and no management plane access is needed for this application. Conversely, if you want a user to be able to read vault properties and tags but not have any access to keys, secrets, or certificates, you can grant this user read access by using RBAC, and no access to the data plane is required.
Store certificates in your key vault.	Azure Resource Manager can securely deploy certificates stored in Azure Key Vault to Azure VMs when the VMs are deployed. By setting appropriate access policies for the key vault, you also control who gets access to your certificate. Another benefit is that you manage all your certificates in one place in Azure Key Vault.
Ensure that you can recover a deletion of key vaults or key vault objects.	Deletion of key vaults or key vault objects can be either inadvertent or malicious. Enable the soft delete and purge protection features of Key Vault, particularly for keys that are used to encrypt data at rest. Deletion of these keys is equivalent to data loss, so you can recover deleted vaults and vault objects if needed. Practice Key Vault recovery operations regularly.

ⓘ **Note**

If a user has contributor permissions (RBAC) to a key vault management plane, they can grant themselves access to the data plane by setting a key vault access policy. It's recommended that you tightly control who has contributor access to your key vaults, to ensure that only authorized persons can access and manage your key vaults, keys, secrets, and certificates.

Next unit: Manage access to secrets, certificates, and keys

Continue >



Manage access to secrets, certificates, and keys

4 minutes

Key Vault access has two facets: the management of the Key Vault itself, and accessing the data contained in the Key Vault. Documentation refers to these facets as the *management plane* and the *data plane*.

These two areas are separated because the creation of the Key Vault (a management operation) is a different role than storing and retrieving a secret stored in the Key Vault. To access a key vault, all users or applications must have proper *authentication* to identify the caller, and *authorization* to determine the operations the caller can perform.

Authentication

Azure Key Vault uses Azure Active Directory to authenticate users and applications that try to access a vault. Authentication is always performed by associated the Azure AD tenant of the subscription that the Key Vault is part of and every user or app making a request must be known to the AAD. There is no support for anonymous access to a Key Vault.

Authorization

Management operations (creating a new Azure Key Vault) use role-based access control (RBAC). There is a built-in role **Key Vault Contributor** that provides access to management features of key vaults, but doesn't allow access to the key vault data. This is the recommended role to use. There's also a **Contributor** role that includes full administration rights - including the ability to grant access to the data plane.

Reading and writing data in the Key Vault uses a separate Key Vault *access policy*. A Key Vault access policy is a permission set assigned to a user or managed identity to read,

write, and/or delete secrets and keys. You can create an access policy using the CLI, REST API, or Azure portal as shown below.

Add access policy

Add access policy

Configure from template (optional)

Key, Secret, & Certificate Management

Key permissions

9 selected

Secret permissions

7 selected

Certificate permissions

15 selected

Select principal

*

Microsoft Learning Partner

>

Authorized application ⓘ

learn-app-dev

>

Add

The system has a list of predefined management options that define the permissions allowed for this policy - here we have **Key, Secret, & Certificate Management** selected which is appropriate to manage secrets in the Key Vault. You can then customize the permissions as desired by changing the **Key permissions** entries. For example, we could adjust the permissions to only allow *read* operations:

Key permissions

2 selected ^

☐ Select all

Key Management Operations

☒ Get

☒ List

☐ Update

☐ Create

☐ Import

☐ Delete

☐ Recover

☐ Backup

☐ Restore

Cryptographic Operations

☐ Decrypt

☐ Encrypt

☐ Unwrap Key

☐ Wrap Key

☐ Verify

☐ Sign

Privileged Key Operations

☐ Purge

Developers will only need `Get` and `List` permissions to a development-environment vault. A lead or senior developer will need full permissions to the vault to change and add secrets when necessary. Full permissions to production-environment vaults are typically reserved for senior operations staff. For applications, often only `Get` permissions are required as they will just need to retrieve secrets.

Restricting network access

Another point to consider with Azure Key Vault is what services in your network can access the vault. In most cases, the network endpoints don't need to be open to the Internet. You should determine the minimum network access required - for example you can restrict Key Vault endpoints to specific Azure Virtual Network subnets, specific IP

addresses, or trusted Microsoft services including Azure SQL, Azure App Service, and various data and storage services that use encryption keys.

keyvault-xtc - Firewalls and virtual networks

Search (Ctrl+J)

Save Discard

Allow access from: ☐ All networks ☒ Selected networks

Configure network access control for your key vault. [Learn More](#)

Virtual networks: [+ Add existing virtual networks](#) [+ Add new virtual network](#)

VIRTUAL NETWORK	SUBNET	RESOURCE GROUP	SUBSCRIPTION
No virtual networks are selected.			

Firewall: ⓘ

IPv4 ADDRESS OR CIDR

IPv4 address or CIDR

Exception:

Allow trusted Microsoft services to bypass this firewall? ☒ Yes ☐ No

This setting is related to firewall only. In order to access this key vault, the trusted service must also be given explicit permissions in the Access policies section.

Next unit: Exercise - store secrets in Azure Key Vault

Continue >



Manage certificates

5 minutes

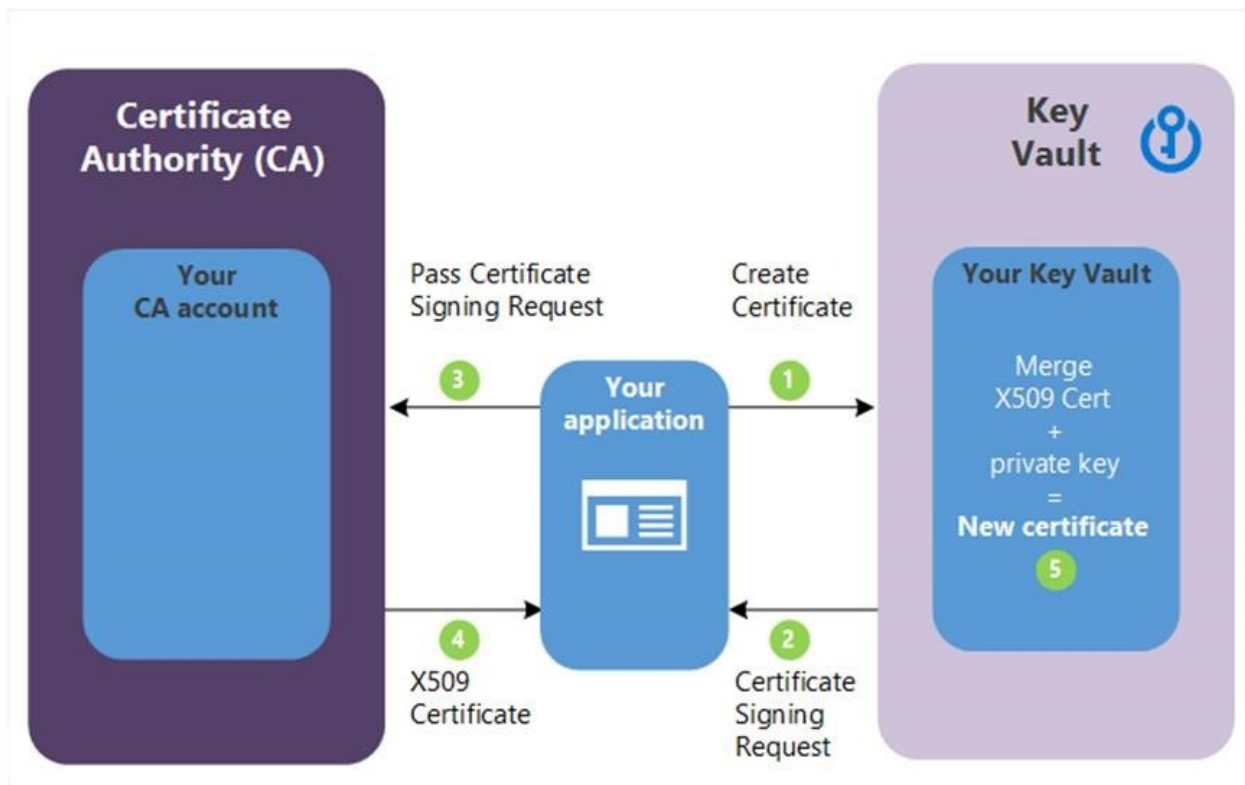
Securely managing certificates is a challenge for every organization. You must ensure that the private key is kept safe, and, as Steve from PetDash found out, certificates have an expiration date and have to be renewed periodically to ensure your website traffic is secure.

Adding certificates to a Key Vault

Azure Key Vault manages X.509 based certificates that can come from several sources.

First, you can create self-signed certificates directly in the Azure portal. This process creates a public/private key pair and signs the certificate with its own key. These certificates can be used for testing and development.

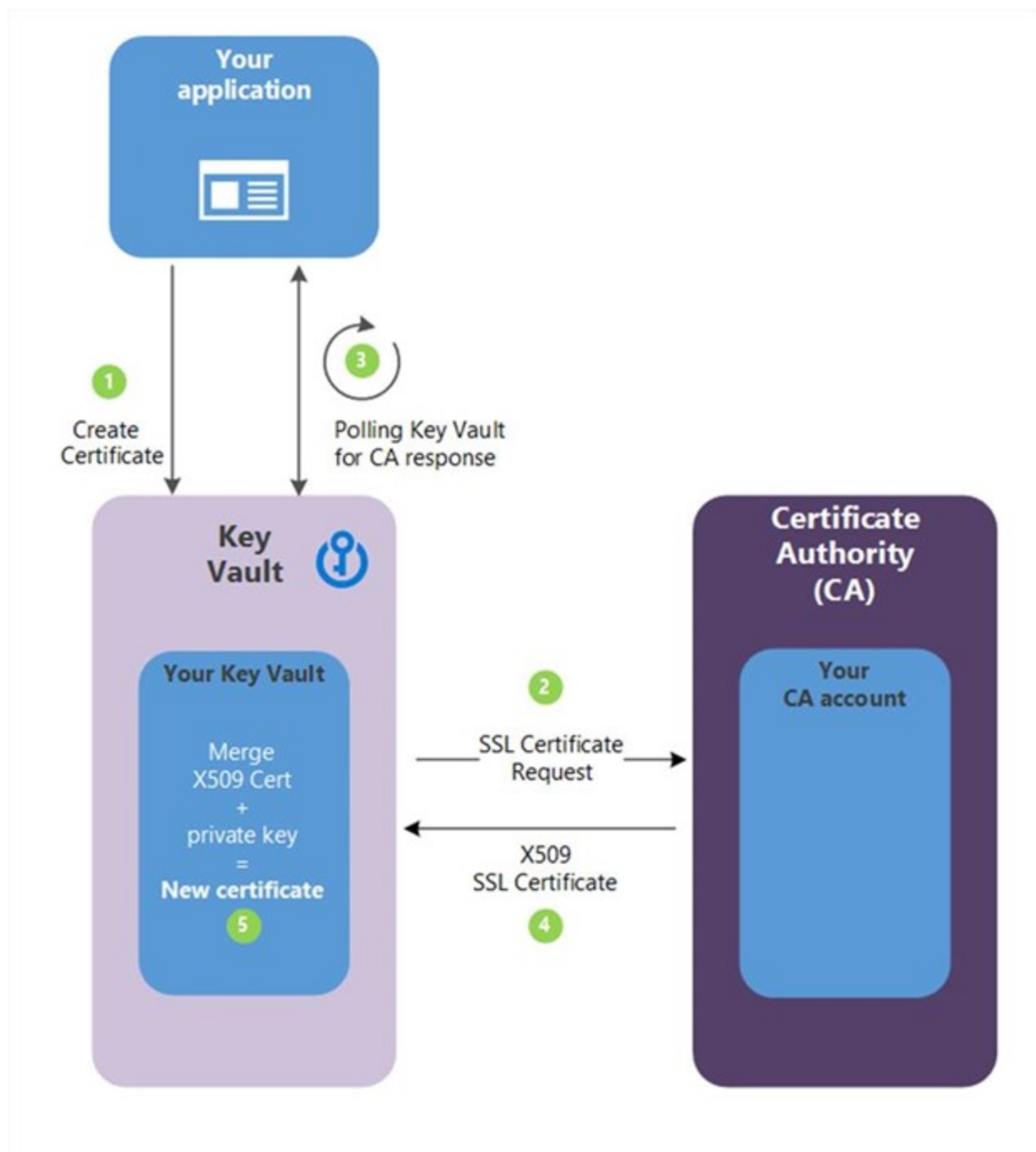
Second, you can create an X.509 certificate signing request (CSR). This creates a public/private key pair in Key Vault along with a CSR you can pass over to your certification authority (CA). The signed X.509 certificate can then be merged with the held key pair to finalize the certificate in Key Vault as shown in the following diagram.



1. In the diagram above, your application is creating a certificate which internally begins by creating a key in your Azure Key Vault.
2. Key Vault returns to your application a Certificate Signing Request (CSR).
3. Your application passes the CSR to your chosen CA.
4. Your chosen CA responds with an X.509 Certificate.
5. Your application completes the new certificate creation with a merger of the X.509 Certificate from your CA.

This approach works with any certificate issuer and provides better security than handling the CSR directly because the private key is created and secured in Azure Key Vault and never revealed.

Third, you can connect your Key Vault with a trusted certificate issuer (referred to as an *integrated CA*) and create the certificate directly in Azure Key Vault. This approach requires a one-time setup to connect the certificate authority. You can then request to create a certificate and the Key Vault will interact directly with the CA to fulfill the request in a similar process to the manual CSR creation process shown above. The full details of this process are presented in the following diagram.



1. In the diagram above, your application is creating a certificate which internally begins by creating a key in your key vault.
2. Key Vault sends a SSL Certificate Request to the CA.
3. Your application polls, in a loop and wait process, for your Key Vault for certificate completion. The certificate creation is complete when Key Vault receives the CA's response with x509 certificate.

4. The CA responds to Key Vault's SSL Certificate Request with an X509 SSL Certificate.
5. Your new certificate creation completes with the merger of the X509 Certificate for the CA.

This approach has several distinct advantages. Because the Key Vault is connected to the issuing CA, it can manage and monitor the lifecycle of the certificate. That means it can automatically renew the certificate, notify you about expiration, and monitor events such as whether the certificate has been revoked.

Finally, you can import existing certificates - this allows you to add certificates to Key Vault that you are already using. The imported certificate can be in either PFX or PEM format and must contain the private key. For example, here's a PowerShell script to upload a certificate:

PowerShell

 Copy

```
$pfxFilePath = "C:\WebsitePrivateCertificate.pfx"
$pwd = "password-goes-here"
$flag = [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable
$pkcs12ContentType = [System.Security.Cryptography.X509Certificates.X509ContentType]::Pkcs12

$collection = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2Collection
$collection.Import($pfxFilePath, $pwd, $flag)


$clearBytes = $collection.Export($pkcs12ContentType)
$fileContentEncoded = [System.Convert]::ToBase64String($clearBytes)
$secret = ConvertTo-SecureString -String $fileContentEncoded -AsPlainText -Force
$secretContentType = 'application/x-pkcs12'





# Replace the <vault-name> and <key-name> below.
Set-AzureKeyVaultSecret -VaultName <vault-name> -Name <key-name> -Secret-Value $secret -ContentType $secretContentType
```


Retrieving certificates from a Key Vault

Once a certificate is stored in your Azure Key Vault, you can use the Azure portal to explore the certificate properties as well as enable or disable a certificate to make it unavailable to clients.

[Home](#) > [Vaultmort - Overview](#) > [Vaultmort - Certificates](#) > [petdash-website](#) > 85de4fc420294f02a752ff57b7a6a63e

 **85de4fc420294f02a752ff57b7a6a63e**
Certificate Version


 Save  Discard  Download in CER format  Download in PFX/PEM format

Properties


Created 9/4/2019, 3:55:43 PM

Updated 9/4/2019, 3:55:43 PM


Certificate Identifier


https://vaultmort.vault.azure.net/certificates/petdash-website/85de4fc420294f02a752ff57b7a6a63e 


Settings

Set activation date?  ☒


Activation Date


09/04/2019  3:45:43 PM

(UTC-05:00) --- Current Time Zone --- 

Set expiration date?  ☒


Expiration Date

09/04/2020  3:55:43 PM

(UTC-05:00) --- Current Time Zone --- 


Enabled? ☒ Yes ☐ No

Tags


0 tags 

Certificate

Subject

CN=petdash.com 

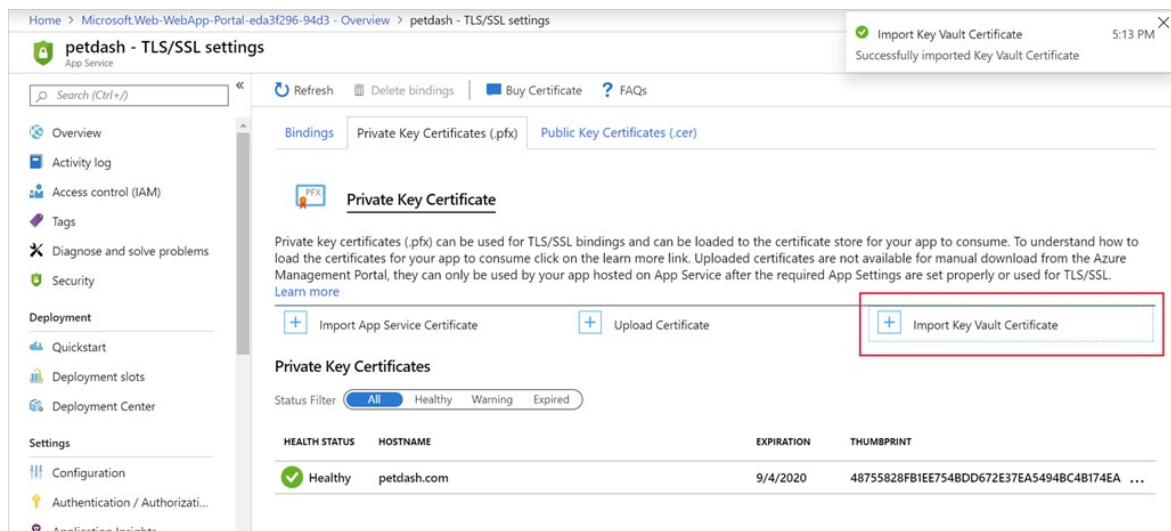
Issuer

CN=petdash.com 

Azure App Service integration

Once you have a public/private key pair certificate in your Azure Key Vault, you can easily associate it to your web app through the Azure portal.

1. Select **TLS/SSL settings** under **Settings**.
2. Select the **Private Key Certificate (.pfx)** tab.
3. Select **+ Import Key Vault Certificate** as shown in the following screenshot.



4. You can then select the vault, which must be in the same subscription, and the secret containing the certificate.

- The certificate must be an X.509 cert with a content type of `application/x-pkcs12` and cannot have a password.

Finally, once the certificate is in place, you'll want to set up a *custom domain*. There's already a built-in certificate for `*.azurewebsites.net`. You can then associate your custom domain with the certificate you've assigned so the server uses your certificate to secure the connection to the browser.

Next unit: Summary

Continue >



Summary

4 minutes

Once you have a key vault, you can start using it to store keys and secrets. Your applications no longer need to persist this confidential data, but can request them from the vault as needed. A key vault allows you to update keys and secrets without affecting the behavior of your application, which opens up a breadth of possibilities for your key, secret, and certificate management.

In this module, you've learned about several security benefits of AKV:

- You can create a segmentation of security roles – no one person has the keys to the kingdom.
- The service is monitored and access is logged – this gives you the capability to track user activity to prevent, detect, and minimize a security incident.
- You can avoid human mistakes – other than the creation of the vault, the services can be automated.
- AKV cloud service are available, accessible, and distributed to ensure high reliability for your services.

Further reading

Continue learning about Azure Key Vault in the Microsoft Learn module [Manage secrets in your server apps with Azure Key Vault](#).

Read Azure Key Vault documentation on topics we covered in this module.

- [What is Azure Key Vault?](#)
 - [Azure Key Vault pricing](#)
 - [Certificate operations](#)
 - [Implementing bring your own key \(BYOK\) for Azure Key Vault](#)
-

Module complete:

Unlock achievement