

Tutorial: Use Azure Key Vault with an Azure web app in .NET

12/21/2018 • 6 minutes to read •  +3

In this article

[Prerequisites](#)

[About Managed Service Identity](#)

[Log in to Azure](#)

[Create a resource group](#)

[Create a key vault](#)

[Add a secret to the key vault](#)

[Create a .NET Core web app](#)

[Open and edit the solution](#)

[Run the web app](#)

[Enable a managed identity](#)

[Assign permissions to your app](#)

[Publish the web app to Azure](#)

[Clean up resources](#)

[Next steps](#)

Azure Key Vault helps you protect secrets such as API keys and database connection strings. It provides you with access to your applications, services, and IT resources.

In this tutorial, you learn how to create an Azure web application that can read information from an Azure key vault. The process uses managed identities for Azure resources. For more information about Azure web applications, see [Azure App Service](#).

The tutorial shows you how to:

- ✓ Create a key vault.
- ✓ Add a secret to the key vault.
- ✓ Retrieve a secret from the key vault.
- ✓ Create an Azure web app.
- ✓ Enable a managed identity for the web app.
- ✓ Assign permission for the web app.
- ✓ Run the web app on Azure.

Before you begin, read [Key Vault basic concepts](#).

If you don't have an Azure subscription, create a [free account](#).

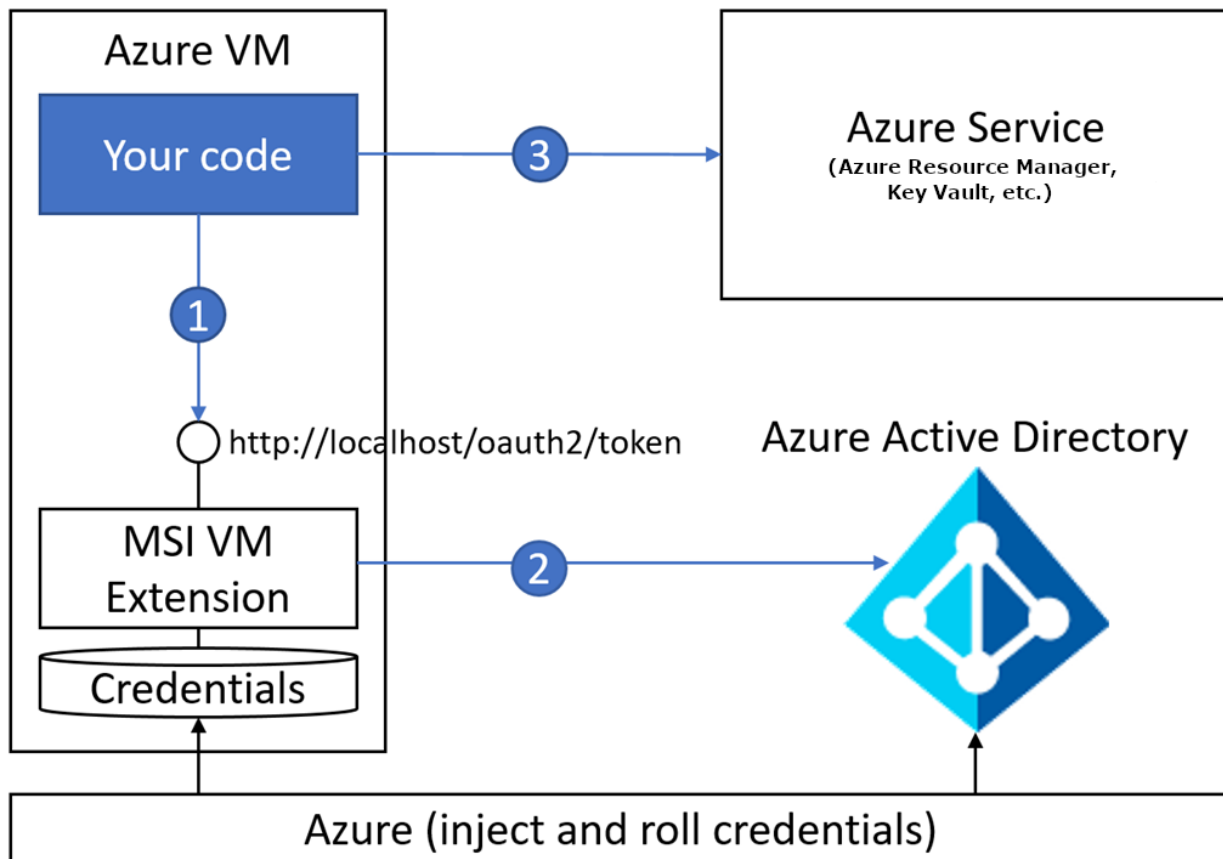
Prerequisites

- For Windows: [.NET Core 2.1 SDK or later](#)
- For Mac: [Visual Studio for Mac](#)
- For Windows, Mac, and Linux:
 - [Git](#)
 - This tutorial requires that you run the Azure CLI locally. You must have the Azure CLI version 2.0.4 or later installed. Run `az --version` to find the version. If you need to install or upgrade the CLI, see [Install Azure CLI 2.0](#).
 - [.NET Core](#)

About Managed Service Identity

Azure Key Vault stores credentials securely, so they're not displayed in your code. However, you need to authenticate to Azure Key Vault to retrieve your keys. To authenticate to Key Vault, you need a credential. It's a classic bootstrap dilemma. Managed Service Identity (MSI) solves this issue by providing a *bootstrap identity* that simplifies the process.

When you enable MSI for an Azure service, such as Azure Virtual Machines, Azure App Service, or Azure Functions, Azure creates a [service principal](#). MSI does this for the instance of the service in Azure Active Directory (Azure AD) and injects the service principal credentials into that instance.



Next, to get an access token, your code calls a local metadata service that's available on the Azure resource. Your code uses the access token that it gets from the local MSI endpoint to authenticate to an Azure Key Vault service.

Log in to Azure

To log in to Azure by using the Azure CLI, enter:

Azure CLI	Copy
<code>az login</code>	

Create a resource group

An Azure resource group is a logical container into which Azure resources are deployed and managed.

Create a resource group by using the [az group create](#) command.

Then, select a resource group name and fill in the placeholder. The following example creates a resource group in the West US location:

Azure CLI	Copy

```
# To list locations: az account list-locations --output table
az group create --name "<YourResourceGroupName>" --location "West US"
```

You use this resource group throughout this tutorial.

Create a key vault

To create a key vault in your resource group, provide the following information:

- Key vault name: a string of 3 to 24 characters that can contain only numbers (0-9), letters (a-z, A-Z), and hyphens (-)
- Resource group name
- Location: **West US**

In the Azure CLI, enter the following command:

Azure CLI



```
az keyvault create --name "<YourKeyVaultName>" --resource-group "
<YourResourceGroupName>" --location "West US"
```

At this point, your Azure account is the only one that's authorized to perform operations on this new vault.

Add a secret to the key vault

Now you can add a secret. It might be a SQL connection string or any other information that you need to keep both secure and available to your application.

To create a secret in the key vault called **AppSecret**, enter the following command:

Azure CLI



```
az keyvault secret set --vault-name "<YourKeyVaultName>" --name "AppSecret"
--value "MySecret"
```

This secret stores the value **MySecret**.

To view the value that's contained in the secret as plain text, enter the following command:

Azure CLI



```
az keyvault secret show --name "AppSecret" --vault-name "<YourKeyVaultName>"
```

This command displays the secret information, including the URI.

After you complete these steps, you should have a URI to a secret in a key vault. Make note of this information for later use in this tutorial.

Create a .NET Core web app


To create a .NET Core web app and publish it to Azure, follow the instructions in [Create an ASP.NET Core web app in Azure](#).

You can also watch this video:



Open and edit the solution

1. Go to the **Pages > About.cshtml.cs** file.
2. Install these NuGet packages:
 - [AppAuthentication](#)
 - [KeyVault](#)
3. Import the following code to the *About.cshtml.cs* file:

C#	 Copy
<pre>using Microsoft.Azure.KeyVault; using Microsoft.Azure.KeyVault.Models; using Microsoft.Azure.Services.AppAuthentication;</pre>	

Your code in the AboutModel class should look like this:

C#

 Copy

```
public class AboutModel : PageModel
{
    public string Message { get; set; }

    public async Task OnGetAsync()
    {
        Message = "Your application description page.";
        int retries = 0;
        bool retry = false;
        try
        {
            /* The next four lines of code show you how to use
            AppAuthentication library to fetch secrets from your key vault */
            AzureServiceTokenProvider azureServiceTokenProvider = new
            AzureServiceTokenProvider();
            KeyVaultClient keyVaultClient = new KeyVaultClient(new
            KeyVaultClient.AuthenticationCallback(azureServiceTokenProvider.KeyVaultTokenCallback));
            var secret = await
            keyVaultClient.GetSecretAsync("https://<YourKeyVaultName>.vault.azure.net/secrets/AppSecret")
            .ConfigureAwait(false);
            Message = secret.Value;
        }
        /* If you have throttling errors see this tutorial
        https://docs.microsoft.com/azure/key-vault/tutorial-net-create-vault-azure-web-app */
        /// <exception cref="KeyVaultErrorException">
        /// Thrown when the operation returned an invalid status code
        /// </exception>
        catch (KeyVaultErrorException keyVaultException)
        {
            Message = keyVaultException.Message;
        }
    }

    // This method implements exponential backoff if there are 429
    errors from Azure Key Vault
    private static long getWaitTime(int retryCount)
    {
        long waitTime = ((long)Math.Pow(2, retryCount) * 100L);
        return waitTime;
    }

    // This method fetches a token from Azure Active Directory, which
    can then be provided to Azure Key Vault to authenticate
    public async Task<string> GetAccessTokenAsync()
    {
        var azureServiceTokenProvider = new
        AzureServiceTokenProvider();
```

```
        string accessToken = await
azureServiceTokenProvider.GetAccessTokenAsync("https://vault.azure.net"
);
        return accessToken;
    }
}
```

Run the web app

1. On the main menu of Visual Studio 2019, select **Debug** > **Start**, with or without debugging.
2. In the browser, go to the **About** page.
The value for **AppSecret** is displayed.

Enable a managed identity

Azure Key Vault provides a way to securely store credentials and other secrets, but your code needs to authenticate to Key Vault to retrieve them. [Managed identities for Azure resources overview](#) helps to solve this problem by giving Azure services an automatically managed identity in Azure AD. You can use this identity to authenticate to any service that supports Azure AD authentication, including Key Vault, without having to display credentials in your code.

In the Azure CLI, to create the identity for this application, run the assign-identity command:

Azure CLI



```
az webapp identity assign --name "<YourAppName>" --resource-group "
<YourResourceGroupName>"
```

Replace <YourAppName> with the name of the published app on Azure.

For example, if your published app name was **MyAwesomeapp.azurewebsites.net**, replace <YourAppName> with **MyAwesomeapp**.

Make a note of the `PrincipalId` when you publish the application to Azure. The output of the command in step 1 should be in the following format:

JSON



```
{
  "principalId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "tenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
}
```

```
"type": "SystemAssigned"
}
```

ⓘ Note

The command in this procedure is the equivalent of going to the [Azure portal](#) and switching the **Identity / System assigned** setting to **On** in the web application properties.

Assign permissions to your app

Replace <YourKeyVaultName> with the name of your key vault, and replace <PrincipalId> with the value of the **PrincipalId** in the following command:

Azure CLI



```
az keyvault set-policy --name '<YourKeyVaultName>' --object-id <PrincipalId>
--secret-permissions get list
```

This command gives the identity (MSI) of the app service permission to do **get** and **list** operations on your key vault.

Publish the web app to Azure

Publish your web app to Azure once again to verify that your live web app can fetch the secret value.

1. In Visual Studio, select the **key-vault-dotnet-core-quickstart** project.
2. Select **Publish** > **Start**.
3. Select **Create**.

When you run the application, you should see that it can retrieve your secret value.

Now, you've successfully created a web app in .NET that stores and fetches its secrets from your key vault.

Clean up resources

When they are no longer needed, you can delete the virtual machine and your key vault.

Next steps

Azure Key Vault Developer's Guide

Is this page helpful?

 Yes  No
