# Lab: Access resource secrets more securely across services

# Student lab answer key

## Microsoft Azure user interface

Given the dynamic nature of Microsoft cloud tools, you might experience Azure UI changes after the development of this training content. These changes might cause the lab instructions and lab steps to not match up.

Microsoft updates this training course when the community brings needed changes to our attention; however, because cloud updates occur frequently, you might encounter UI changes before this training content updates. **If this occurs, adapt to the changes, and then work through them in the labs as needed.**

## Instructions

### Before you start

#### Sign in to the lab virtual machine

Sign in to your Windows 10 virtual machine by using the following credentials:

* Username: **Admin**
* Password: **Pa55w.rd**

**Note**: Instructions to connect to the virtual lab environment will be provided by your instructor.

#### Review the installed applications

Find the taskbar on your Windows 10 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

* Microsoft Edge
* File Explorer

### Exercise 1: Create Azure resources

#### Task 1: Open the Azure portal

1. On the taskbar, select the **Microsoft Edge** icon.
2. In the open browser window, go to the **Azure portal** (<https://portal.azure.com>).
3. Enter the email address for your Microsoft account, and then select **Next**.
4. Enter the **password** for your Microsoft account, and then select **Sign in**.

* **Note**: If this is your first time signing in to the Azure portal, you will be offered a tour of the portal. Select **Get Started** to skip the tour and begin using the portal.

#### Task 2: Create an Azure Storage account

1. In the Azure portal's navigation pane, select **All services**.
2. From the **All services** blade, select **Storage Accounts**.
3. From the **Storage accounts** blade, find your list of Storage instances.
4. From the **Storage accounts** blade, select **Add**.
5. Find the tabs from the **Create storage account** blade, such as **Basics**.

* **Note**: Each tab represents a step in the workflow to create a new storage account. You can select **Review + Create** at any time to skip the remaining tabs.

1. From the **Basics** tab, perform the following actions:
   1. Leave the **Subscription** text box set to its default value.
   2. In the **Resource group** section, select **Create new**, enter **SecureFunction**, and then select **OK**.
   3. In the **Storage account name** text box, enter **securestor*[yourname]***.
   4. In the **Location** drop-down list, select the **(US) East US** region.
   5. In the **Performance** section, select **Standard**.
   6. In the **Account kind** drop-down list, select **StorageV2 (general purpose v2)**.
   7. In the **Replication** drop-down list, select **Locally-redundant storage (LRS)**.
   8. In the **Access tier** section, ensure that **Hot** is selected.
   9. Select **Review + Create**.
2. From the **Review + Create** tab, review the options that you selected during the previous steps.
3. Select **Create** to create the storage account by using your specified configuration.

* **Note**: Wait for the creation task to complete before you move forward with this lab.

1. In the Azure portal's navigation pane, select **All services**.
2. From the **All services** blade, select **Storage Accounts**.
3. From the **Storage accounts** blade, select the storage account instance with the prefix *securestor\**.
4. From the **Storage account** blade, find the **Settings** section, and then select the **Access keys** link.
5. From the **Access keys** blade, select any one of the keys and record the value in either of the **Connection string** boxes. You'll use this value later in this lab.

* **Note**: It doesn't matter which connection string you choose. They are interchangeable.

#### Task 3: Create an Azure Key Vault

1. In the Azure portal's navigation pane, select the **Create a resource** link.
2. From the **New** blade, find the **Search the Marketplace** text box above the list of featured services.
3. In the search box, enter **Vault**, and then select Enter.
4. From the **Marketplace** search results blade, select the **Key Vault** result.
5. From the **Key Vault** blade, select **Create**.
6. Find the tabs from the **Create key vault** blade, such as **Basics**.

* **Note**: Each tab represents a step in the workflow to create a new key vault. You can select **Review + Create** at any time to skip the remaining tabs.

1. From the **Basics** tab, perform the following actions:
   1. Leave the **Subscription** text box set to its default value.
   2. In the **Resource group** section, select **Use existing**, and then select **SecureFunction** in the list.
   3. In the **Key vault name** text box, enter **securevault*[yourname]***.
   4. In the **Region** drop-down list, select the **East US** region.
   5. In the **Pricing tier** drop-down list, select **Standard**.
   6. Select **Review + Create**.
2. From the **Review + Create** tab, review the options that you selected during the previous steps.
3. Select **Create** to create the key vault by using your specified configuration.

* **Note**: Wait for the creation task to complete before you move forward with this lab.

#### Task 4: Create an Azure Functions app

1. In the Azure portal's navigation pane, select the **Create a resource** link.
2. From the **New** blade, find the **Search the Marketplace** text box above the list of featured services.
3. In the search box, enter **Function**, and then select Enter.
4. From the **Marketplace** search results blade, select the **Function App** result.
5. From the **Function App** blade, select **Create**.
6. Find the tabs from the **Function App** blade, such as **Basics**.

* **Note**: Each tab represents a step in the workflow to create a new function app. You can select **Review + Create** at any time to skip the remaining tabs.

1. From the **Basics** tab, perform the following actions:
   1. Leave the **Subscription** text box set to its default value.
   2. In the **Resource group** section, select **Use existing**, and then select **SecureFunction** in the list.
   3. In the **Function app name** text box, enter **securefunc*[yourname]***.
   4. In the **Publish** section, select **Code**.
   5. In the **Runtime stack** drop-down list, select **.NET Core**.
   6. In the **Version** drop-down list, select **3.1**.
   7. In the **Region** drop-down list, select the **East US** region.
   8. Select **Next: Hosting**.
2. From the **Hosting** tab, perform the following actions:
   1. In the **Storage account** drop-down list, select the **securestor*[yourname]*** storage account that you created earlier in this lab.
   2. In the **Operating System** section, select **Windows**.
   3. In the **Plan type** drop-down list, select the **Consumption (Serverless)** option.
   4. Select **Next: Monitoring**.
3. From the **Monitoring** tab, perform the following actions:
   1. In the **Enable Application Insights** section, select **No**.
   2. Select **Review + Create**.
4. From the **Review + Create** tab, review the options that you selected during the previous steps.
5. Select **Create** to create the function app by using your specified configuration.

* **Note**: Wait for the creation task to complete before you move forward with this lab.

#### Review

In this exercise, you created all the resources that you'll use for this lab.

### Exercise 2: Configure secrets and identities

#### Task 1: Configure a system-assigned managed service identity

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securefunc*[yourname]*** function app that you created earlier in this lab.
4. From the **Function Apps** blade, select the **Platform features** tab.
5. From the **Platform features** tab, select the **Identity** link in the **Networking** section.
6. From the **Identity** blade, find the **System assigned** tab, and then perform the following actions:
   1. In the **Status** section, select **On**, and then select **Save**.
   2. In the confirmation dialog box, select **Yes**.

* **Note**: Wait for the system-assigned managed identity to be created before you move forward with this lab.

#### Task 2: Create a Key Vault secret

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securevault*[yourname]*** key vault that you created earlier in this lab.
4. From the **Key Vault** blade, select the **Secrets** link in the **Settings** section.
5. In the Secrets pane, select **Generate/Import**.
6. From the **Create a secret** blade, perform the following actions:
   1. In the **Upload options** drop-down list, select **Manual**.
   2. In the **Name** text box, enter **storagecredentials**.
   3. In the **Value** text box, enter the storage account connection string that you recorded earlier in this lab.
   4. Leave the **Content Type** text box set to its default value.
   5. Leave the **Set activation date** text box set to its default value.
   6. Leave the **Set expiration date** text box set to its default value.
   7. In the **Enabled** section, select **Yes**, and then select **Create**.

* **Note**: Wait for the secret to be created before you move forward with this lab.

1. Back in the Secrets pane, select the **storagecredentials** item in the list.
2. In the Versions pane, select the latest version of the **storagecredentials** secret.
3. In the Secret Version pane, perform the following actions:
   1. Find the metadata for the latest version of the secret.
   2. Select **Show secret value** to find the value of the secret.
   3. Record the value of the **Secret Identifier** text box because you'll use this later in the lab.

* **Note**: You are recording the value of the **Secret Identifier** text box, not the **Secret Value** text box.

#### Task 3: Configure a Key Vault access policy

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securevault*[yourname]*** key vault that you created earlier in this lab.
4. From the **Key Vault** blade, select the **Access policies** link in the **Settings** section.
5. In the Access policies pane, select **Add Access Policy**.
6. From the **Add access policy** blade, perform the following actions:
   1. Select the **Select principal** link.
   2. From the **Principal** blade, find and then select the service principal named **securefunc*[yourname]***, and then select **Select**.
   * **Note**: The system-assigned managed identity you created earlier in this lab will have the same name as the Azure Function resource.
   1. Leave the **Key permissions** list set to its default value.
   2. In the **Secret permissions** drop-down list, select the **GET** permission.
   3. Leave the **Certificate permissions** list set to its default value.
   4. Leave the **Authorized application** text box set to its default value.
   5. Select **Add**.
7. Back in the Access policies pane, select **Save**.

* **Note**: Wait for your changes to the access policies to save before you move forward with this lab.

#### Review

In this exercise, you created a server-assigned managed service identity for your function app and then gave that identity the appropriate permissions to get the value of a secret in your key vault. Finally, you created a secret that you'll use within your function app.

### Exercise 3: Write function app code

#### Task 1: Create a Key Vault-derived application setting

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securefunc*[yourname]*** function app that you created earlier in this lab.
4. From the **Function Apps** blade, select the **Platform features** tab.
5. From the **Platform features** tab, select the **Configuration** link in the **General Settings** section.
6. In the **Configuration** section, perform the following actions:
   1. Select the **Application settings** tab, and then select **New application setting**.
   2. In the **Add/Edit application setting** pop-up window, in the **Name** text box, enter **StorageConnectionString**.
   3. In the **Value** text box, construct a value by using the following syntax: **@Microsoft.KeyVault(SecretUri=*Secret Identifier*)**
   * **Note**: You'll need to build a reference to your ***Secret Identifier*** by using the above syntax. For example, if your secret identifier is **https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf**, then your value would be **@Microsoft.KeyVault(SecretUri=https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf)**
   1. Leave the **deployment slot setting** text box set to its default value.
   2. Select **OK** to close the pop-up window and return to the **Configuration** section.
   3. Select **Save** from the blade to persist your settings.
   4. In the **Save Changes** confirmation popup dialog, select **Continue**.

* **Note**: Wait for your application settings to persist before you move forward with the lab.

#### Task 2: Create an HTTP-triggered function

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securefunc*[yourname]*** function app that you created earlier in this lab.
4. From the **Function App** blade, select the plus sign (**+**) next to the **Functions** drop-down list.
5. In the **New Azure Function** quickstart, perform the following actions:
   1. Under the **Choose a Development Environment** header, select **In-Portal**, and then select **Continue**.
   2. Under the **Create a Function** header, select **More templates**, and then select **Finish and view templates**.
   3. In the list of templates, select **HTTP trigger**.
   4. In the **New Function** pop-up window, find the **Name** text box, and then enter **FileParser**.
   5. In the **New Function** pop-up window, find the **Authorization level** list, and then select **Anonymous**.
   6. In the **New Function** pop-up window, select **Create**.
6. In the function editor, find the example function script:

* #r "Newtonsoft.Json"  
    
  using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
  using Microsoft.Extensions.Primitives;  
  using Newtonsoft.Json;  
    
  public static async Task<IActionResult> Run(HttpRequest req, ILogger log)  
  {  
    
   log.LogInformation("C# HTTP trigger function processed a request.");  
    
   string name = req.Query["name"];  
    
   string requestBody = await new StreamReader(req.Body).ReadToEndAsync();  
    
   dynamic data = JsonConvert.DeserializeObject(requestBody);  
    
   name = name ?? data?.name;  
    
   return name != null  
   ? (ActionResult)new OkObjectResult($"Hello, {name}")  
   : new BadRequestObjectResult("Please pass a name from the query string or in the request body");  
  }

1. **Delete** all the example code, and then in the function editor, copy and paste the following placeholder function:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
   return new OkObjectResult("Test Successful");  
  }

1. Select **Save and run** to save the script and perform a test of the function.
2. The Test and Logs panes will automatically open when the script runs for the first time.

* **Note**: You may notice warning content during compilation in the log. This can be safely ignored.

1. Find the **Output** text box in the Test pane. You should now notice the value **Test Successful** returned from the function.

#### Task 3: Test the Key Vault-derived application setting

1. Delete the existing code within the **Run** method of the script.
2. Observe the **Run** method, which should now include:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
    
  }

1. Add the following line of code to get the value of the **StorageConnectionString** application setting by using the **Environment.GetEnvironmentVariable** method:

* string connectionString = Environment.GetEnvironmentVariable("StorageConnectionString");

1. Add the following line of code to return the value of the *connectionString* variable by using the **OkObjectResult** class constructor:

* return new OkObjectResult(connectionString);

1. Observe the **Run** method, which should now include:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
   string connectionString = Environment.GetEnvironmentVariable("StorageConnectionString");  
   return new OkObjectResult(connectionString);  
  }

1. Select **Save and run** to save the script and perform a test of the function.
2. Find the **Output** text box in the Test pane. You should now notice the connection string returned from the function.

* **Note**: You may notice a warning message. This is simply a C# compiler warning that says you are using an asynchronous method without asynchronous code. This warning will no longer show up later in the lab.

#### Review

In this exercise, you used a service identity to read the value of a secret stored in Key Vault and returned that value as the result of an function app.

### Exercise 4: Access storage account blobs

#### Task 1: Upload a sample storage blob

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securestor*[yourname]*** storage account that you created earlier in this lab.
4. From the **Storage account** blade, select the **Containers** link in the **Blob service** section.
5. In the **Containers** section, select **+ Container**.
6. In the **New container** pop-up window, perform the following actions:
   1. In the **Name** text box, enter **drop**.
   2. In the **Public access level** drop-down list, select **Blob (anonymous read access for blobs only)**, and then select **OK**.
7. Back in the **Containers** section, select the newly created **drop** container.
8. From the **Container** blade, select **Upload**.
9. In the **Upload blob** pop-up window, perform the following actions:
   1. In the **Files** section, select the **Folder** icon.
   2. In the **File Explorer** window, browse to **Allfiles (F):\Allfiles\Labs\07\Starter**, select the **records.json** file, and then select **Open**.
   3. Ensure that **Overwrite if files already exist** is selected, and then select **Upload**.

* **Note**: Wait for the blob to upload before you continue with this lab.

1. Back from the **Container** blade, select the **records.json** blob in the list of blobs.
2. From the **Blob** blade, find the blob metadata, and then copy the URL for the blob.
3. On the taskbar, right-click the **Microsoft Edge** icon or activate the shortcut menu, and then select **New window**.
4. In the new browser window, go to the URL that you copied for the blob.
5. You should now notice the JavaScript Object Notation (JSON) contents of the blob. Close the browser window with the JSON contents.
6. Return to the browser window with the Azure portal, and then close the **Blob** blade.
7. Back from the **Container** blade, select **Change access level policy**.
8. In the **Change access level** pop-up window, perform the following actions:
   1. In the **Public access level** drop-down list, select **Private (no anonymous access)**.
   2. Select **OK**.
9. On the taskbar, right-click the **Microsoft Edge** icon or activate the shortcut menu, and then select **New window**.
10. In the new browser window, go to the URL that you copied for the blob.
11. You should now notice an error message indicating that the resource wasn't found.

* **Note**: If you don't notice the error message, your browser might have cached the file. Press Ctrl+F5 to refresh the page until you notice the error message.

#### Task 2: Pull the storage account SDK from NuGet

1. In the Azure portal's navigation pane, select the **Resource groups** link.
2. From the **Resource groups** blade, find and then select the **SecureFunction** resource group that you created earlier in this lab.
3. From the **SecureFunction** blade, select the **securefunc*[yourname]*** function app that you created earlier in this lab.
4. From the **Function App** blade, find and then select the existing **FileParser** function to open the editor for the function.

* **Note**: You might need to expand the **Functions** option in the menu of the blade.

1. In the editor, select **View files** to open the tab.
2. From the **View files** tab, select **Add**.
3. In the **File name** dialog box, enter **function.proj**, and then select Enter, which displays an empty code editor.
4. In the editor, insert this configuration content:

* <Project Sdk="Microsoft.NET.Sdk">  
   <PropertyGroup>  
   <TargetFramework>netstandard2.0</TargetFramework>  
   </PropertyGroup>  
   <ItemGroup>  
   <PackageReference Include="Azure.Storage.Blobs" Version="12.4.0" />  
   </ItemGroup>  
  </Project>

1. In the editor, select **Save** to persist your configuration changes.

* **Note**: This .proj file contains the NuGet package reference necessary to import the [Azure.Storage.Blobs](https://www.nuget.org/packages/Azure.Storage.Blobs/12.4.0) package.

1. Select the **run.csx** file to return to the editor for the **FileParser** function.
2. Minimize the **View files** tab.

* **Note**: You can minimize the tab by selecting the arrow associated with the tab header.

1. Within the editor, delete the existing code in the **Run** method of the script.
2. In the code file, add the following line of code to create a **using** directive for the **Azure.Storage** namespace:

* using Azure.Storage;

1. In the code file, add the following line of code to create a **using** directive for the **Azure.Storage.Blobs** namespace:

* using Azure.Storage.Blobs;

1. Add the following line of code to create a **using** directive for the **Azure.Storage.Blobs.Models** namespace:

* using Azure.Storage.Blobs.Models;

1. Observe the **Run** method, which should now include:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
  using Azure.Storage;  
  using Azure.Storage.Blobs;  
  using Azure.Storage.Blobs.Models;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
    
  }

#### Task 3: Write storage account code

1. Add the following line of code in the **Run** method to get the value of the **StorageConnectionString** application setting by using the **Environment.GetEnvironmentVariable** method:

* string connectionString = Environment.GetEnvironmentVariable("StorageConnectionString");

1. Add the following line of code to create a new instance of the **BlobServiceClient** class by passing in your *connectionString* variable to the constructor:

* BlobServiceClient serviceClient = new BlobServiceClient(connectionString);

1. Add the following line of code to use the **BlobServiceClient.GetBlobContainerClient** method, while passing in the **drop** container name to create a new instance of the **BlobContainerClient** class that references the container that you created earlier in this lab:

* BlobContainerClient containerClient = serviceClient.GetBlobContainerClient("drop");

1. Add the following line of code to use the **BlobContainerClient.GetBlobClient** method, while passing in the **records.json** blob name to create a new instance of the **BlobClient** class that references the blob that you uploaded earlier in this lab:

* BlobClient blobClient = containerClient.GetBlobClient("records.json");

1. Observe the **Run** method, which should now include:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
  using Azure.Storage;  
  using Azure.Storage.Blobs;  
  using Azure.Storage.Blobs.Models;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
   string connectionString = Environment.GetEnvironmentVariable("StorageConnectionString");  
   BlobServiceClient serviceClient = new BlobServiceClient(connectionString);  
   BlobContainerClient containerClient = serviceClient.GetBlobContainerClient("drop");  
   BlobClient blobClient = containerClient.GetBlobClient("records.json");  
  }

#### Task 4: Download a blob

1. Add the following line of code to use the **BlobClient.DownloadAsync** method to download the contents of the referenced blob asynchronously and store the result in a variable named *response*:

* var response = await blobClient.DownloadAsync();

1. Add the following line of code to return the various content stored in the *content* variable by using the **FileStreamResult** class constructor:

* return new FileStreamResult(response?.Value?.Content, response?.Value?.ContentType);

1. Observe the **Run** method, which should now include:

* using System.Net;  
  using Microsoft.AspNetCore.Mvc;  
  using Azure.Storage;  
  using Azure.Storage.Blobs;  
  using Azure.Storage.Blobs.Models;  
    
  public static async Task<IActionResult> Run(HttpRequest req)  
  {  
   string connectionString = Environment.GetEnvironmentVariable("StorageConnectionString");  
   BlobServiceClient serviceClient = new BlobServiceClient(connectionString);  
   BlobContainerClient containerClient = serviceClient.GetBlobContainerClient("drop");  
   BlobClient blobClient = containerClient.GetBlobClient("records.json");  
   var response = await blobClient.DownloadAsync();  
   return new FileStreamResult(response?.Value?.Content, response?.Value?.ContentType);  
  }

1. Select **Save and run** to save the script and perform a test of the function.
2. Find the **Output** text box in the Test pane. You should now notice the content of the **$/drop/records.json** blob stored in your storage account.

#### Review

In this exercise, you used C# code to access a storage account, and then download the contents of a blob.

### Exercise 5: Clean up your subscription

#### Task 1: Open Azure Cloud Shell and list resource groups

1. In the Azure portal's navigation pane, select the **Cloud Shell** icon to open a new shell instance.

* **Note**: The **Cloud Shell** icon is represented by a greater than sign () and underscore character (\_).

1. If this is your first time opening Cloud Shell using your subscription, you can use the **Welcome to Azure Cloud Shell Wizard** to configure Cloud Shell for first-time usage. Perform the following actions in the wizard:
   1. A dialog box prompts you to create a new storage account to begin using the shell. Accept the default settings, and then select **Create storage**.

* **Note**: Wait for Cloud Shell to finish its initial setup procedures before moving forward with the lab. If you don't notice Cloud Shell configuration options, this is most likely because you are using an existing subscription with this course's labs. The labs are written with the presumption that you are using a new subscription.

1. At the **Cloud Shell** command prompt in the portal, enter the following command, and then select Enter to list all resource groups in the subscription:

* az group list

1. At the command prompt, enter the following command, and then select Enter to find a list of possible commands to delete a resource group:

* az group delete --help

#### Task 2: Delete a resource group

1. At the command prompt, enter the following command, and then select Enter to delete the **SecureFunction** resource group:

* az group delete --name SecureFunction --no-wait --yes

1. Close the Cloud Shell pane in the portal.

#### Task 3: Close the active application

1. the currently running Microsoft Edge application.

#### Review

In this exercise, you cleaned up your subscription by removing the resource groups used in this lab.

©2019 Microsoft Corporation. All rights reserved. The text in this document is available under the [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/legalcode), additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are ***not*** included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.