1.3 - Secure Azure AI services

Overview

Securing Azure AI services can help prevent data loss and privacy violations for user data that may be a part of the solution.

After completing this module, you will know how to:

- Consider authentication for Azure AI services
- Manage network security for Azure AI services

Introduction

Azure AI services provides multiple layers of security that you should consider when implementing a solution.

After completing this module, you'll learn how to:

- Consider authentication for Azure AI services.
- Manage network security for Azure AI services.

Consider authentication

By default, access to Azure AI services resources is restricted by using subscription keys. Management of access to these keys is a primary consideration for security.

Regenerate keys

You should regenerate keys regularly to protect against the risk of keys being shared with or accessed by unauthorized users. You can regenerate keys using the Azure portal, or using the `az cognitiveservices account keys regenerate` Azure command-line interface (CLI) command.

Each AI service is provided with two keys, enabling you to regenerate keys without service interruption. To accomplish this:

1. If you're using both keys in production, change your code so that only one key is in use. For example, configure all production applications to use key 1.
2. Regenerate key 2.
3. Switch all production applications to use the newly regenerated key 2.
4. Regenerate key 1
5. Finally, update your production code to use the new key 1.

For example, to regenerate keys in the Azure portal, you can do the following:

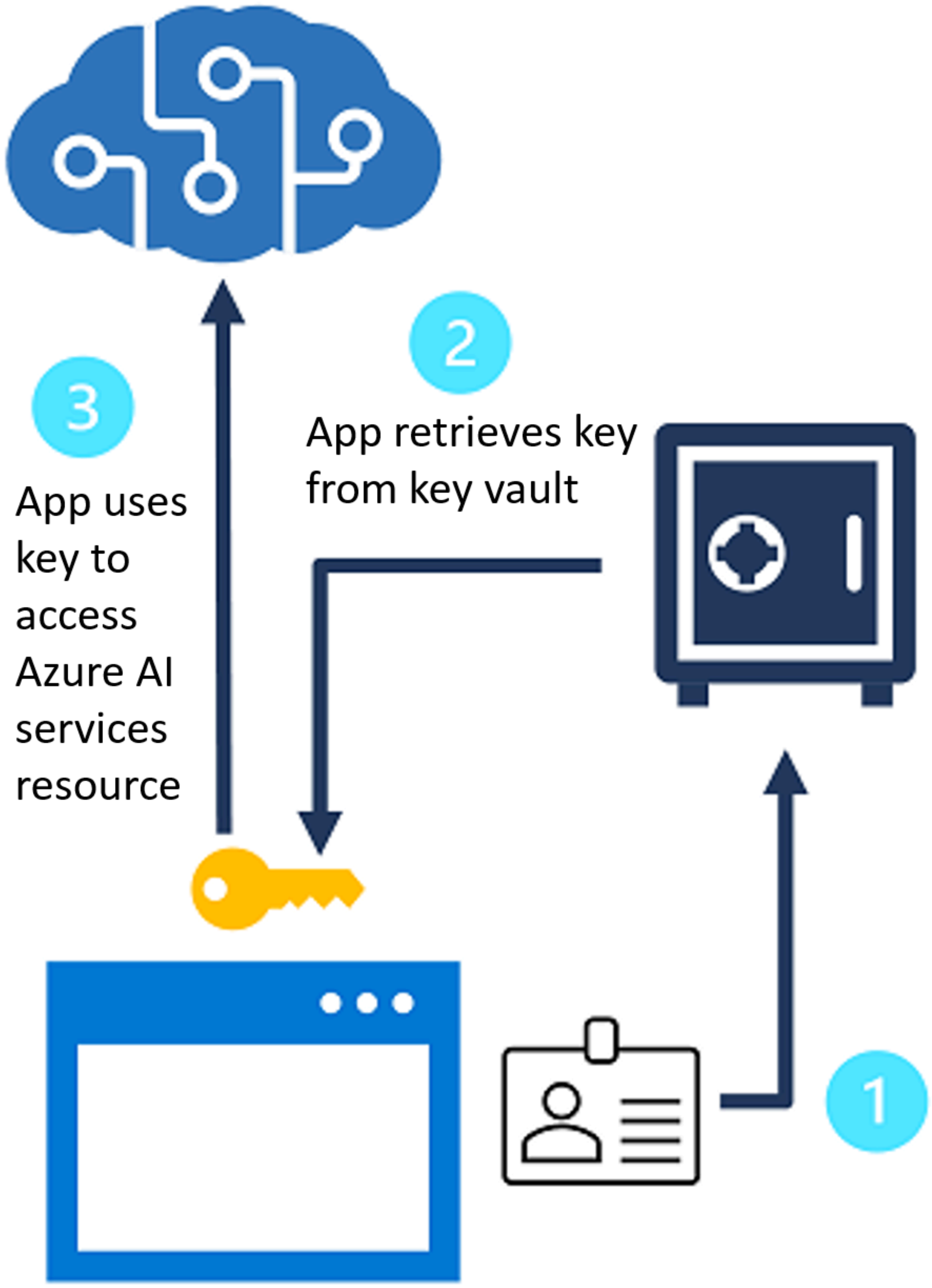
1. In the Azure portal, go to your resource's Keys and Endpoint pane.
2. Then select **Regenerate Key1** or select **Regenerate Key2**, depending on which one you want to regenerate at the time.

Protect keys with Azure Key Vault

Azure Key Vault is an Azure service in which you can securely store secrets (such as passwords and keys). Access to the key vault is granted to *security principals*, which you can think of user identities that are authenticated using Microsoft Entra ID.

Administrators can assign a security principal to an application (in which case it is known as a *service principal*) to define a *managed identity* for the application. The application can then use this identity to access the key vault and retrieve a secret to which it has access. Controlling access to the secret in this way minimizes the risk of it being compromised by being hard-coded in an application or saved in a configuration file.

You can **store the subscription keys for an AI services resource in Azure Key Vault, and assign a managed identity to client applications that need to use the service**. The applications can then retrieve the key as needed from the key vault, without risk of exposing it to unauthorized users.


3
App uses key to access Azure AI services resource
2
App retrieves key from key vault
1

Token-based authentication

When using the REST interface, some AI services support (or even *require*) token-based authentication. In these cases, the subscription key is presented in an initial request to obtain an authentication token, which has a valid period of 10 minutes. Subsequent requests must present the token to validate that the caller has been authenticated.

> Tip: When using an SDK, the calls to obtain and present a token are handled for you by the SDK.

Microsoft Entra ID authentication

Azure AI services supports Microsoft Entra ID authentication, enabling you to **grant access to specific service principals or managed identities** for apps and services running in Azure.

Microsoft Entra ID is a cloud-based identity and access management solution.

> Note: For more information about authentication options for AI services, see the AI services documentation.

There are different ways you can authenticate against Azure AI services using Microsoft Entra ID, including:

(1) Authenticate using service principals

The overall process to authenticate against Azure AI services using service principals is as follows:

(i) Create a custom subdomain

You can create a custom subdomain in different ways including through the **Azure portal, Azure CLI, or PowerShell**. For example, you can create a subdomain using PowerShell in the Azure Cloud Shell. To do this, you select your subscription using the following command:

```
Set-AzContext -SubscriptionName <Your-Subscription-Name>
```

Then, you create your Azure AI services resource specifying a custom subdomain by running the following:

```
$account = New-AzCognitiveServicesAccount -ResourceGroupName <your-resource-group-name>
-name <your-account-name> -Type <your-account-type> -SkuName <your-sku-type> -Location
<your-region> -CustomSubdomainName <your-unique-subdomain-name>
```

Once created, your subdomain name will be returned in the response.

(ii) Assign a role to a service principal

You've created an Azure AI resource that is linked with a custom subdomain. Next, you assign a role to a service principal.

To start, you'll need to register an application. To do this, you run the following command:

```
$SecureStringPassword = ConvertTo-SecureString -String <your-password> -AsPlainText -
Force

$app = New-AzureADApplication -DisplayName <your-app-display-name> -IdentifierUris
<your-app-uris> -PasswordCredentials $SecureStringPassword
```

This creates the application resource. Then you use the **New-AzADServicePrincipal** command to create a service principal and provide your application's ID:

```
New-AzADServicePrincipal -ApplicationId <app-id>
```

Finally, you assign the **Cognitive Services Users** role to your service principal by running:

```
New-AzRoleAssignment -ObjectId <your-service-principal-object-id> -Scope <account-id> -
RoleDefinitionName "Cognitive Services User"
```

(2) Authenticate using managed identities

Managed identities come in two types:

- **System-assigned managed identity**: A managed identity is created and **linked to a specific resource**, such as a virtual machine that needs to access Azure AI services. When the resource is deleted, the identity is deleted as well.
- **User-assigned managed identity**: The managed identity is created to be useable by multiple resources instead of being tied to one. It exists independently of any single resource.

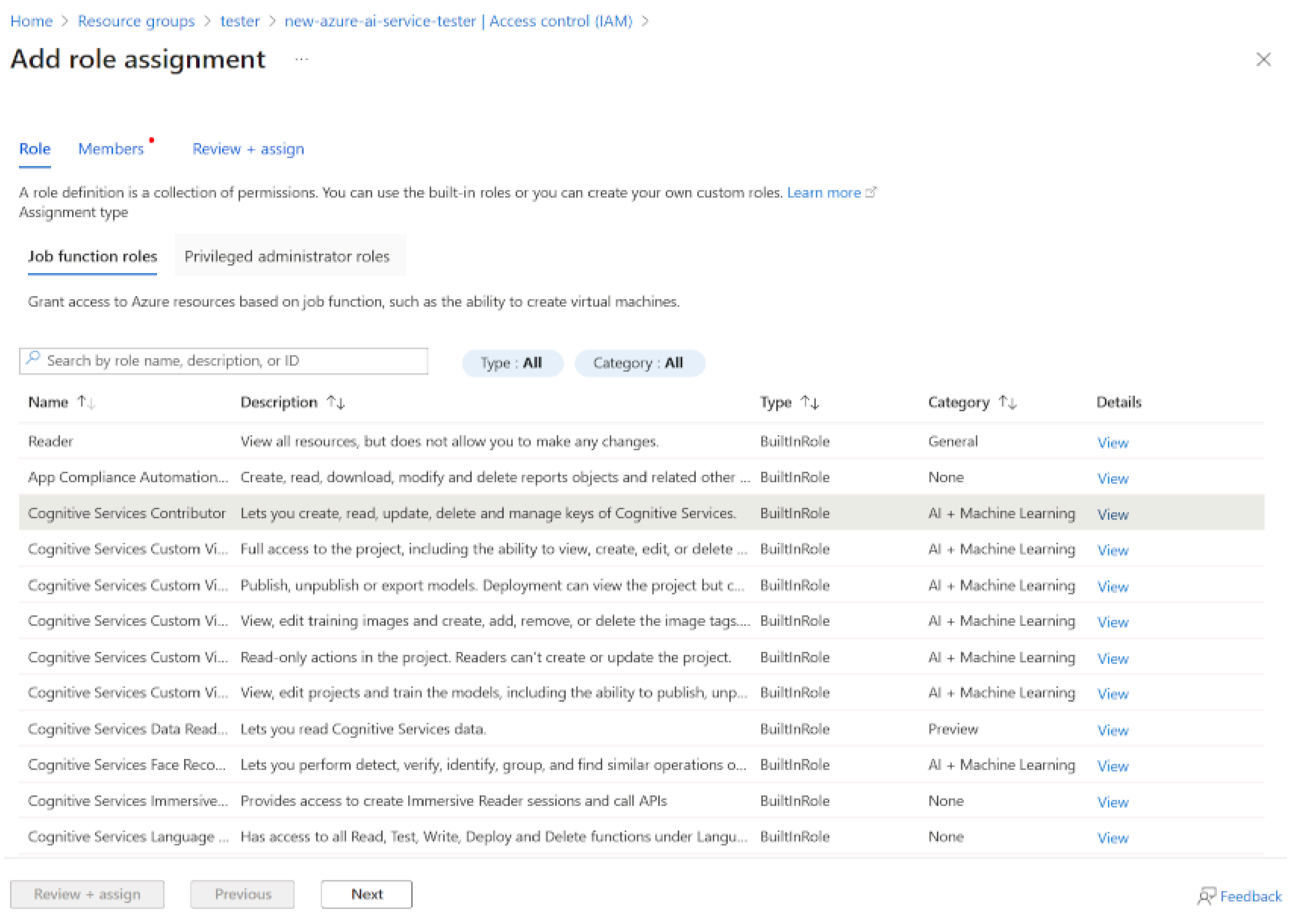
You can assign each type of managed identity to a resource either during creation of the resource, or after it has already been created.

For example, suppose you have a virtual machine in Azure that you intend to use for daily access to Azure AI services. To enable a system-assigned identity for this virtual machine, first you make sure your Azure account has the Virtual Machine Contributor role. Then you can run the following command using Azure CLI in the Azure Cloud Shell terminal:

```
az vm identity assign -g <my-resource-group> -n <my-vm>
```

Then you can grant access to Azure AI services in the Azure portal using the following:

1. Go to the Azure AI services resource you want to grant the virtual machine's managed identity access.
2. In the overview panel, select **Access control (IAM)**.
3. Select **Add**, and then select **Add role assignment**.
4. In the Role tab, select **Cognitive Services Contributor**.


Home > Resource groups > tester > new-azure-ai-service-tester | Access control (IAM) >

Add role assignment

Role | Members | Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more
Assignment type

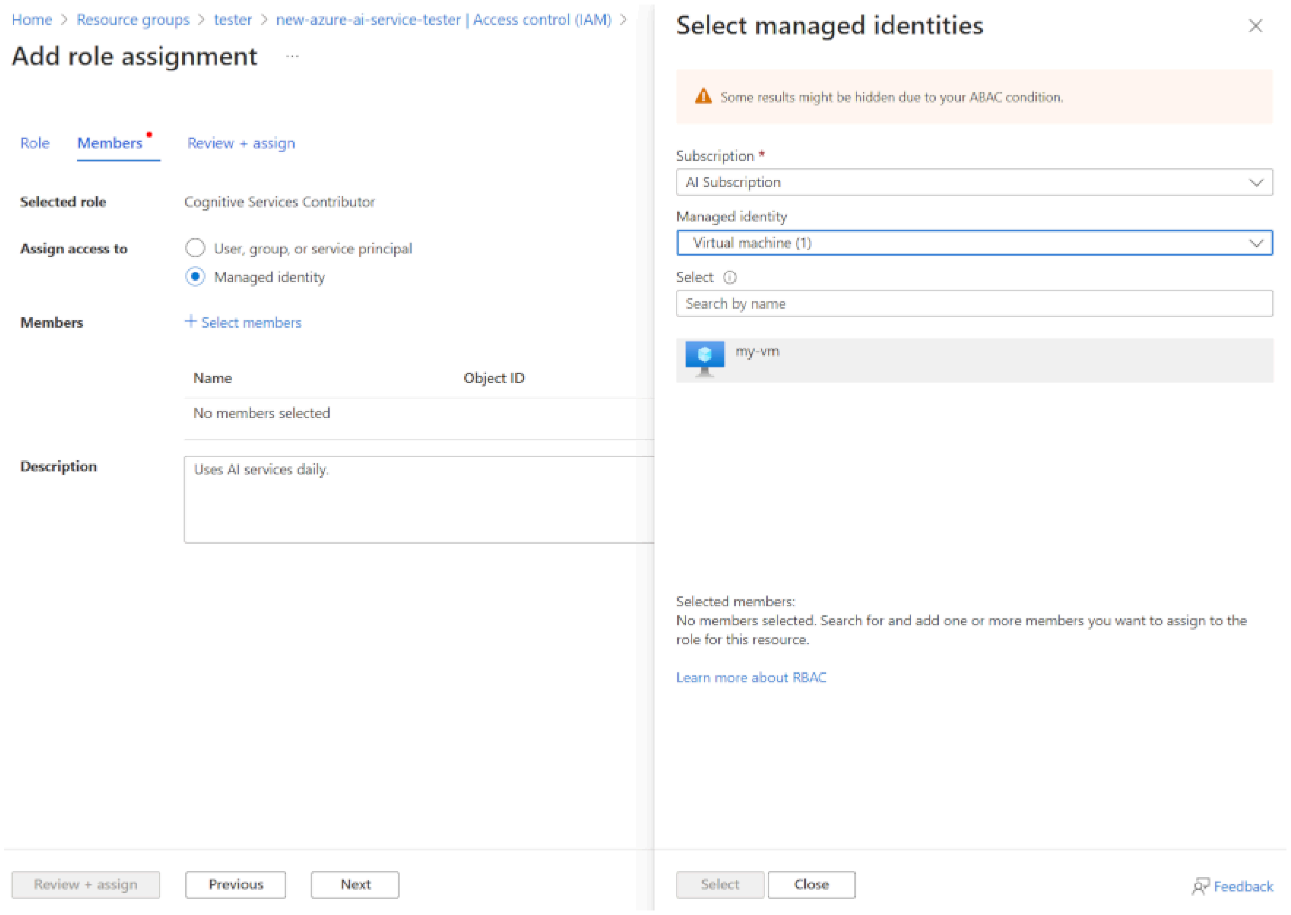
Job function roles | Privileged administrator roles

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

Search by role name, description, or ID | Type : All | Category : All

Name	Description	Type	Category	Details
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	View
App Compliance Automation...	Create, read, download, modify and delete reports objects and related other ...	BuiltInRole	None	View
Cognitive Services Contributor	Lets you create, read, update, delete and manage keys of Cognitive Services.	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vi...	Full access to the project, including the ability to view, create, edit, or delete ...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vi...	Publish, unpublish or export models. Deployment can view the project but c...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vi...	View, edit training images and create, add, remove, or delete the image tags....	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vi...	Read-only actions in the project. Readers can't create or update the project.	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vi...	View, edit projects and train the models, including the ability to publish, unp...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Data Read...	Lets you read Cognitive Services data.	BuiltInRole	Preview	View
Cognitive Services Face Reco...	Lets you perform detect, verify, identify, group, and find similar operations o...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Immersive...	Provides access to create Immersive Reader sessions and call APIs	BuiltInRole	None	View
Cognitive Services Language ...	Has access to all Read, Test, Write, Deploy and Delete functions under Langu...	BuiltInRole	None	View

5. In the Members tab, for the Assign access to, select **Managed identity**. Then, select **+ Select members**.


Home > Resource groups > tester > new-azure-ai-service-tester | Access control (IAM) >

Add role assignment

Role | Members | Review + assign

Selected role: Cognitive Services Contributor

Assign access to: User, group, or service principal [✓ Managed identity]

Members: + Select members

Name	Object ID
No members selected	

Description: Uses AI services daily.

Select managed identities

Some results might be hidden due to your ABAC condition.

Subscription *: AI Subscription

Managed identity: Virtual machine (1)

Select: Search by name

my-vm

Selected members:
No members selected. Search for and add one or more members you want to assign to the role for this resource.

Learn more about RBAC

6. Ensure that your subscription is selected in the Subscription dropdown. And for Managed identity, select **Virtual machine**.
7. Select your virtual machine in the list, and select **Select**.

8. Finally, select **Review + assign** to review, and then **Review + assign** again to finish.

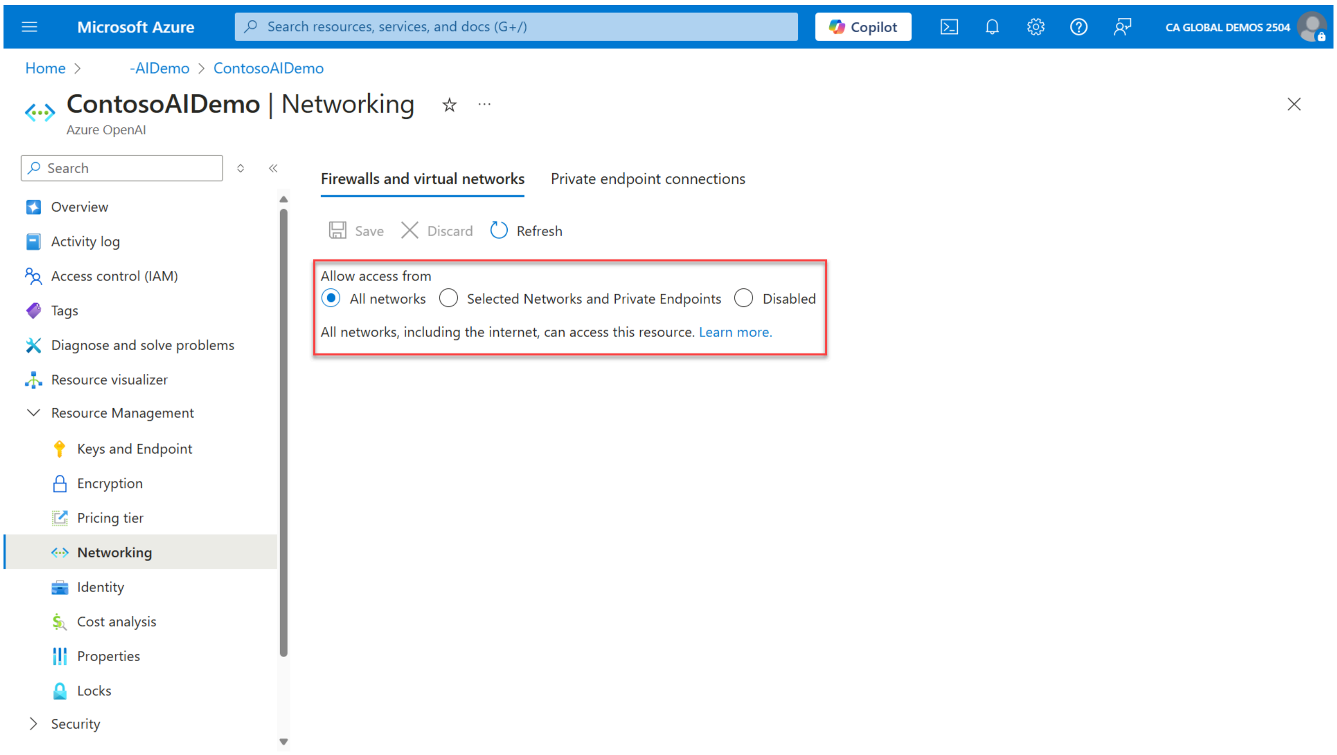
> Note: For more details on how to configure managed identities including user-managed identities, see Configure managed identities for Azure resource on Azure VM using Azure CLI

Implement network security

Network security is an important measure to ensure unauthorized users can't reach the services that you are protecting. Limiting what users can see is always a great idea, since they can't compromise what they can't see.

Apply network access restrictions

By default, Azure AI services are accessible from all networks. Some individual AI services resources (such as **Azure AI Face service**, **Azure AI Vision**, and others) can be **configured to restrict access to specific network addresses** - **either public Internet addresses or addresses on virtual networks**.



Allow access from
[✓ All networks] Selected Networks and Private Endpoints Disabled
All networks, including the internet, can access this resource. Learn more.

With network restrictions enabled, a client **trying to connect from an IP address that isn't allowed** will receive an **Access Denied** error.

> Note: For more information about configuring network access for AI services, see the AI services documentation.

Exercise - Manage Azure AI Services Security

Security is a critical consideration for any application, and as a developer you should ensure that access to resources such as Azure AI services is restricted to only those who require it.

Access to Azure AI services is typically controlled through authentication keys, which are generated when you initially create an Azure AI services resource.

Clone the repository in Visual Studio Code

You'll develop your code using Visual Studio Code. The code files for your app have been provided in a GitHub repo.

> **Tip**: If you have already cloned the **mslearn-ai-services** repo recently, open it in Visual Studio code. Otherwise, follow these steps to clone it to your development environment.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/mslearn-ai-services` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo, if necessary

   > **Note**: If you are prompted to add required assets to build and debug, select **Not Now**.

5. Expand the `Labfiles/02-ai-services-security` folder.

Provision an Azure AI Services resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI Services** resource.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. In the top search bar, search for *Azure AI services*, select **Azure AI Services**, and create an Azure AI services multi-service account resource with the following settings:
   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group - use the one provided)*
   - **Region**: *Choose any available region*
   - **Name**: *Enter a unique name*
   - **Pricing tier**: Standard S0
3. Select the required checkboxes and create the resource.
4. Wait for deployment to complete, and then view the deployment details.

Manage authentication keys

When you created your Azure AI services resource, two authentication keys were generated. You can manage these in the Azure portal or by using the Azure command line interface (CLI).

1. In the Azure portal, go to your Azure AI services resource and view its **Keys and Endpoint** page. This page contains the information that you will need to connect to your resource and use it from applications you develop. Specifically:
   - An HTTP *endpoint* to which client applications can send requests.
   - Two *keys* that can be used for authentication (client applications can use either of the keys. A common practice is to use one for development, and another for production. You can easily

regenerate the development key after developers have finished their work to prevent continued access).
   - The *location* where the resource is hosted. This is required for requests to some (but not all) APIs.
2. Now you can use the following command to get the list of Azure AI services keys, replacing `<resourceName>` with the name of your Azure AI services resource, and `<resourceGroup>` with the name of the resource group in which you created it.

   ```
   az cognitiveservices account keys list --name <resourceName> --resource-group
   <resourceGroup>
   ```

   The command returns a list of the keys for your Azure AI services resource - there are two keys, named **key1** and **key2**.
3. To test your Azure AI service, you can use **curl** - a command line tool for HTTP requests. In the **02-ai-services-security** folder, open **rest-test.cmd** and edit the **curl** command it contains (shown below), replacing `<yourEndpoint>` and `<yourKey>` with your endpoint URI and **Key1** key to use the Analyze Text API in your Azure AI services resource.

   `curl -X POST "<yourEndpoint>/language/:analyze-text?api-version=2023-04-01" -H "Content-Type: application/json" -H "Ocp-Apim-Subscription-Key: 81468b6728294aab99c489664a818197" --data-ascii "{'analysisInput':{'documents':[{'id':1,'text':'hello'}]}, 'kind': 'LanguageDetection'}"`
4. Save your changes, and then run the following command:
   `bash ./rest-test.cmd`
   The command returns a JSON document containing information about the language detected in the input data (which should be English).
5. If a key becomes compromised, or the developers who have it no longer require access, you can regenerate it in the portal or by using the Azure CLI. Run the following command to regenerate your **key1** key (replacing _ `resourceName>` and `<resourceGroup>` for your resource).

   ```
   az cognitiveservices account keys regenerate --name <resourceName> --resource
   group <resourceGroup> --key-name key1
   ```

The list of keys for your Azure AI services resource is returned - note that **key1** has changed since you last retrieved them.

6. Re-run the **rest-test** command with the old key (you can use the **^** arrow on your keyboard to cycle through previous commands), and verify that it now fails.
7. Edit the *curl* command in **rest-test.cmd** replacing the key with the new **key1** value, and save the changes. Then rerun the **rest-test** command and verify that it succeeds.

> **Tip**: In this exercise, you used the full names of Azure CLI parameters, such as **--resource-group**. You can also use shorter alternatives, such as **-g**, to make your commands less verbose (but a little harder to understand). The Azure AI Services CLI command reference lists the parameter options for each Azure AI services CLI command.

Secure key access with Azure Key Vault

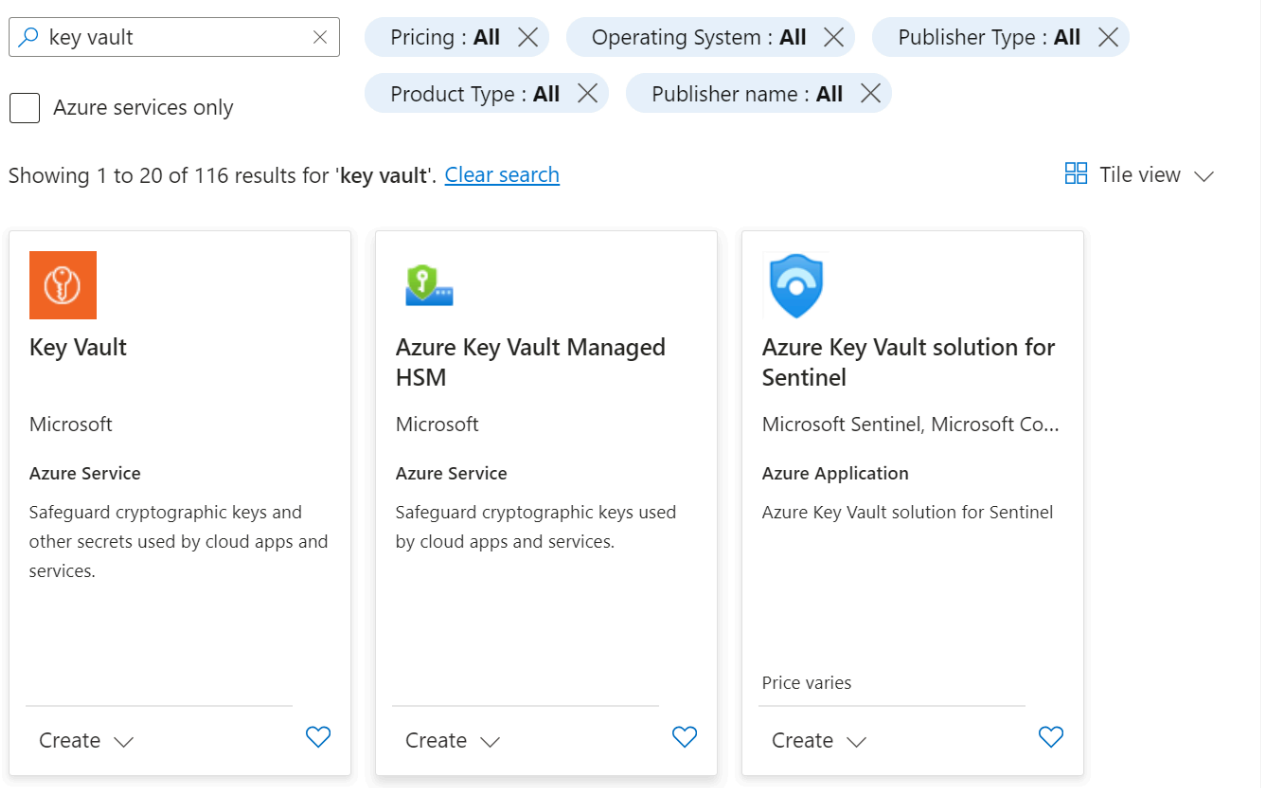
You can develop applications that consume Azure AI services by using a key for authentication. However, this means that the application code must be able to obtain the key.

One option is to store the key in an environment variable or a configuration file where the application is deployed, but this approach leaves the key vulnerable to unauthorized access. A better approach when developing applications on Azure is to store the key securely in Azure Key Vault, and provide access to the key through a *managed identity* (in other words, a user account used by the application itself).

(i) Create a key vault and add a secret

First, you need to create a key vault and add a *secret* for the Azure AI services key.

1. Make a note of the **key1** value for your Azure AI services resource (or copy it to the clipboard).
2. In the Azure portal, on the **Home** page, select the **＋Create a resource** button, search for *Key Vault*



Showing 1 to 20 of 116 results for 'key vault'. Clear search

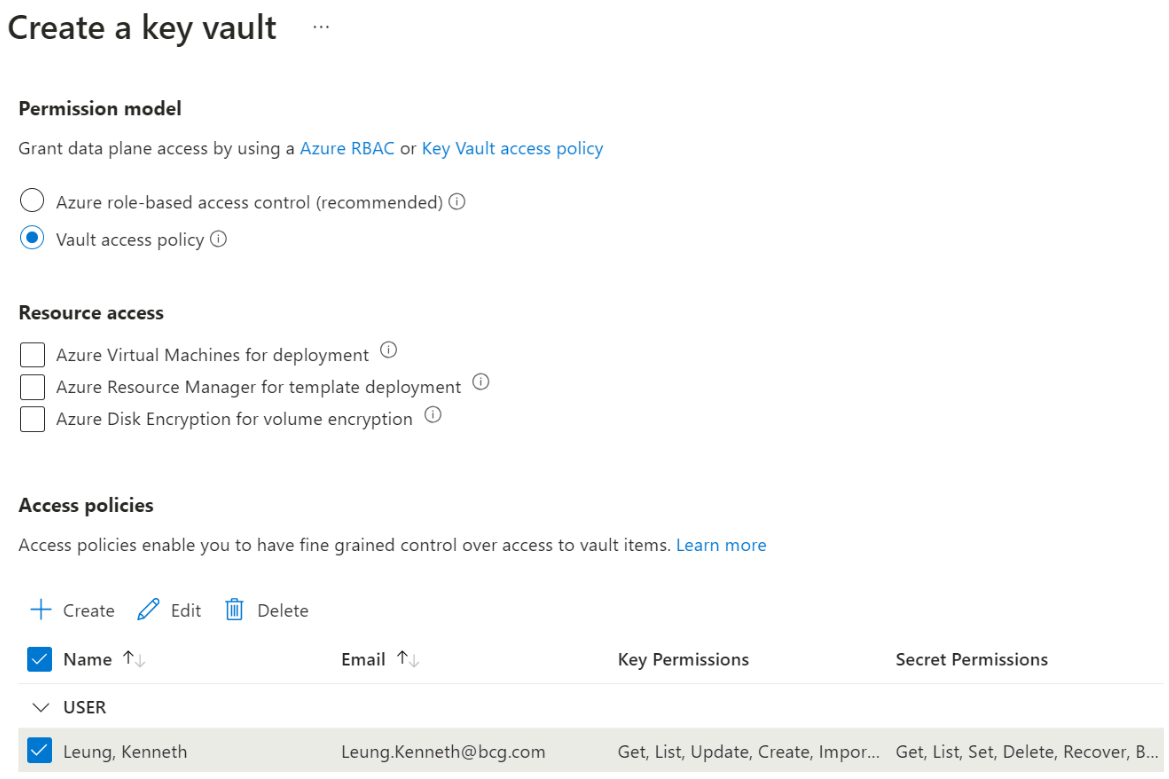
Key Vault
Microsoft
Azure Service
Safeguard cryptographic keys and other secrets used by cloud apps and services.

Azure Key Vault Managed HSM
Microsoft
Azure Service
Safeguard cryptographic keys used by cloud apps and services.

Azure Key Vault solution for Sentinel
Microsoft Sentinel, Microsoft Co...
Azure Application
Azure Key Vault solution for Sentinel
Price varies

3. Create a **Key Vault** resource with the following settings:
   - **Basics** tab
     - **Subscription**: *Your Azure subscription*
     - **Resource group**: *The same resource group as your Azure AI service resource*
     - **Key vault name**: *Enter a unique name*
     - **Region**: *The same region as your Azure AI service resource*
     - **Pricing tier**: Standard
   - **Access configuration** tab
     - **Permission model**: Vault access policy (Key Vault access policy determines whether a given security principal, namely a user, application or user group, can perform different operations on keys, secrets and certificates.)

- Scroll down to **Access policies** section and select your user using the checkbox on the left.



Create a key vault

Permission model

Grant data plane access by using a Azure RBAC or Key Vault access policy

- ○ Azure role-based access control (recommended)
- ● Vault access policy

Resource access

- ☐ Azure Virtual Machines for deployment
- ☐ Azure Resource Manager for template deployment
- ☐ Azure Disk Encryption for volume encryption

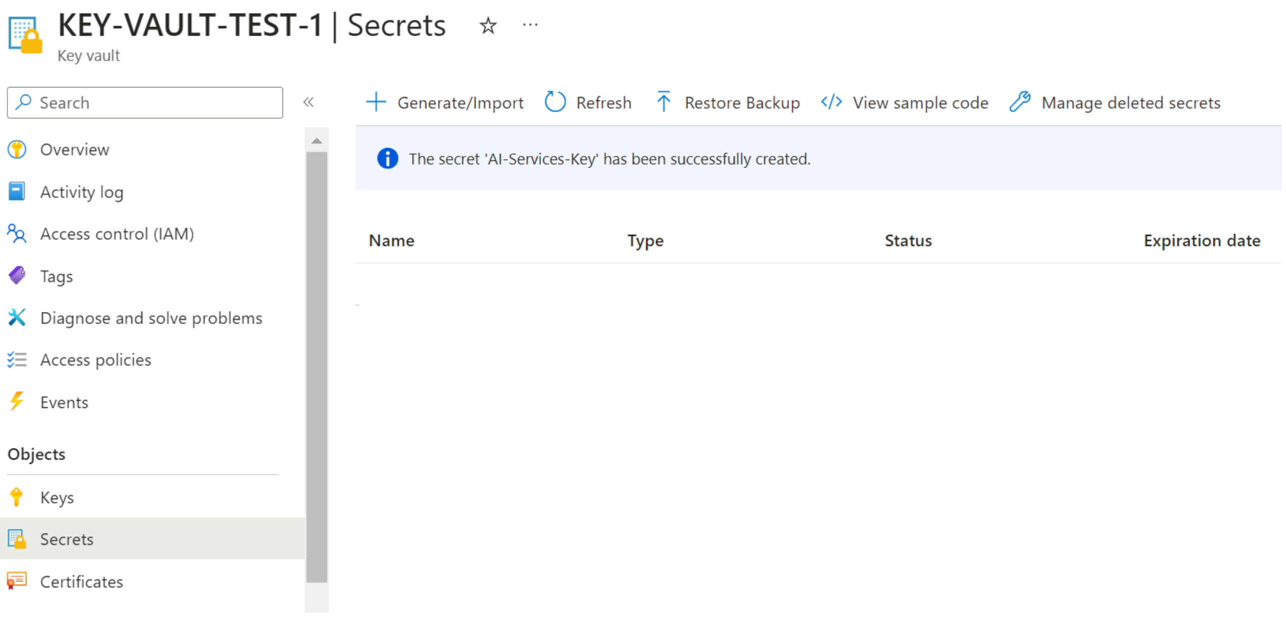
Access policies

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

Name	Email	Key Permissions	Secret Permissions
USER			
✓ Leung, Kenneth	Leung.Kenneth@bcg.com	Get, List, Update, Create, Impor...	Get, List, Set, Delete, Recover, B...

- Then select **Review + create**, and select **Create** to create your resource.

4. Wait for deployment to complete and then go to your key vault resource.
5. In the left navigation pane, select **Secrets** (in the Objects section).



KEY-VAULT-TEST-1 | Secrets

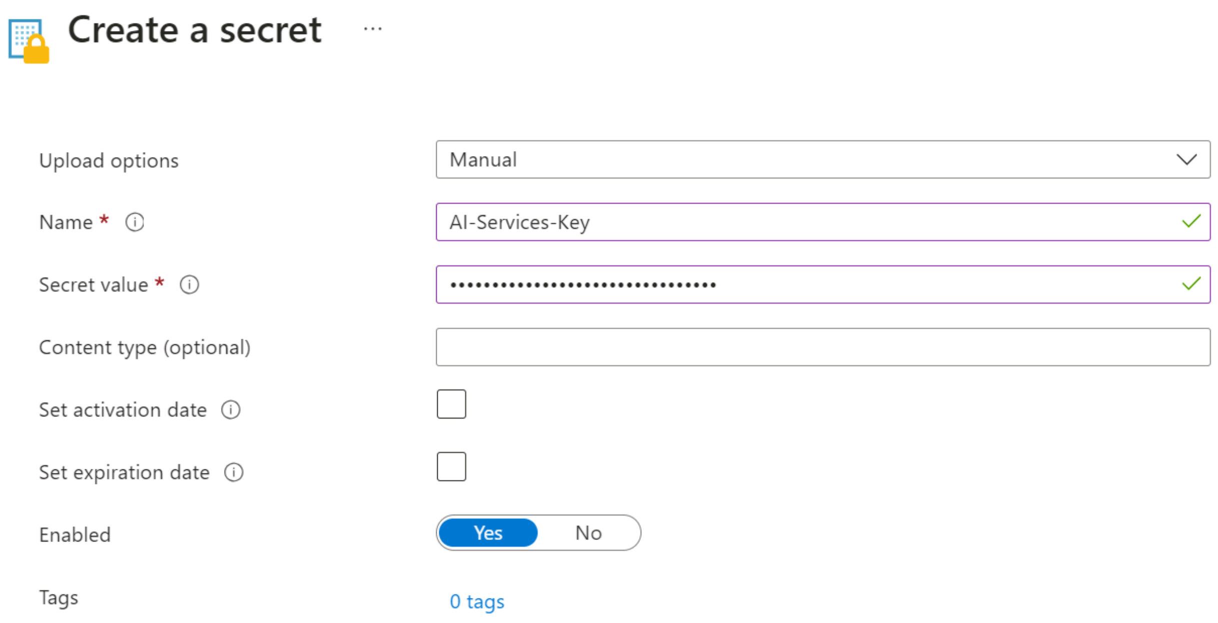
Key vault

The secret 'AI-Services-Key' has been successfully created.

Name	Type	Status	Expiration date

6. Select **+ Generate/Import** and add a new secret with the following settings :
   - **Upload options**: Manual
   - **Name**: AI-Services-Key *(it's important to match this exactly, because later you'll run code that retrieves the secret based on this name)*

- **Value**: *Your **key1** Azure AI services key*



Create a secret

Upload options	Manual
Name *	AI-Services-Key
Secret value *	••••••••••••••••••••••••••••••••
Content type (optional)	
Set activation date	☐
Set expiration date	☐
Enabled	[✓ Yes] No
Tags	0 tags

7. Select **Create**.

(ii) Create a service principal

To access the secret in the key vault, your application must use a service principal that has access to the secret.

> Recap: An Azure service principal is **an identity created for use with applications, hosted services, and automated tools to access Azure resources**. This access is restricted by the roles assigned to the service principal, giving you control over which resources can be accessed and at which level.

You'll use the Azure command line interface (CLI) to create the service principal, find its object ID, and grant access to the secret in Azure Vault.

1. Run the following Azure CLI command, replacing `<spName>` with a unique suitable name for an application identity (for example, *ai-app* with your initials appended on the end; the name must be unique within your tenant).
   Also replace `<subscriptionId>` and `<resourceGroup>` with the correct values for your subscription ID and the resource group containing your Azure AI services and key vault resources:

   ```
   > **Tip**: If you are unsure of your subscription ID, use the **az account
   show** command to retrieve your subscription information - the subscription ID is
   the **id** attribute in the output. If you see an error about the object already
   existing, please choose a different unique name.

   ```
   az ad sp create-for-rbac -n "api://<spName>" --role owner --scopes
   subscriptions/<subscriptionId>/resourceGroups/<resourceGroup>
   ```
   ```

   The output of this command includes information about your new service principal. It should look similar to this:

```
{
    "appId": "abcd12345efghi67890jklmn",
    "displayName": "api://ai-app-",
    "password": "1a2b3c4d5e6f7g8h9i0j",
    "tenant": "1234abcd5678fghi90jklm"
}
```

Make a note of the **appId**, **password**, and **tenant** values - you will need them later (if you close this terminal, you won't be able to retrieve the password; so it's important to note the values now - you can paste the output into a new text file on your local machine to ensure you can find the values you need later!)

2. To get the **object ID** of your service principal, run the following Azure CLI command, replacing `<appId>` with the value of your service principal's app ID.

   ```
   az ad sp show --id <appId>
   ```

3. Copy the `id` value in the JSON returned in response.
4. To assign permission for your new service principal to access secrets in your Key Vault, run the following Azure CLI command, replacing `<keyVaultName>` with the name of your Azure Key Vault resource and `<objectId>` with the value of your service principal's ID value you've just copied.

   ```
   az keyvault set-policy -n <keyVaultName> --object-id <objectId> --secret
   permissions get list
   ```

(iii) Use the service principal in an application

Now you're ready to use the service principal identity in an application, so it can access the secret Azure AI services key in your key vault and use it to connect to your Azure AI services resource.

> **Note**: In this exercise, we'll store the service principal credentials in the application configuration and use them to authenticate a **ClientSecretCredential** identity in your application code. This is fine for development and testing, but in a real production application, an administrator would assign a *managed identity* to the application so that it uses the service principal identity to access resources, without caching or storing the password.

1. In your terminal, switch to the **C-Sharp** or **Python** folder depending on your language preference by running `cd C-Sharp` or `cd Python`. Then run `cd keyvault_client` to navigate to the app folder.
2. Install the packages you will need to use for Azure Key Vault and the Text Analytics API in your Azure AI services resource by running the appropriate command for your language preference:

   C#

   ```
   dotnet add package Azure.AI.TextAnalytics --version 5.3.0
   dotnet add package Azure.Identity --version 1.5.0
   dotnet add package Azure.Security.KeyVault.Secrets --version 4.2.0-beta.3
   ```

   Python

```
pip install azure-ai-textanalytics==5.3.0
pip install azure-identity==1.5.0
pip install azure-keyvault-secrets==4.2.0
```

3. View the contents of the **keyvault-client** folder, and note that it contains a file for configuration settings:

   - **C#**: `appsettings.json`
   - **Python**: `.env`

   Open the configuration file and update the configuration values it contains to reflect the following settings:
   - The **endpoint** for your Azure AI Services resource
   - The name of your **Azure Key Vault** resource
   - The **tenant** for your service principal
   - The **appId** for your service principal
   - The **password** for your service principal
     Save your changes by pressing **CTRL+S**.
4. Note that the **keyvault-client** folder contains a code file for the client application:

   - **C#**: `Program.cs`
   - **Python**: `keyvault-client.py`

     > View here: keyvault-client.py

   Open the code file and review the code it contains, noting the following details:
   - The namespace for the SDK you installed is imported
   - Code in the **Main** function r**etrieves the application configuration settings, and then it uses the service principal credentials to get the Azure AI services key from the key vault.**
   - The **GetLanguage** function uses the SDK to create a client for the service, and then uses the client to detect the language of the text that was entered.
5. Enter the following command to run the program:
   C#

   ```
   dotnet run
   ```

   Python

   ```
   python keyvault-client.py
   ```

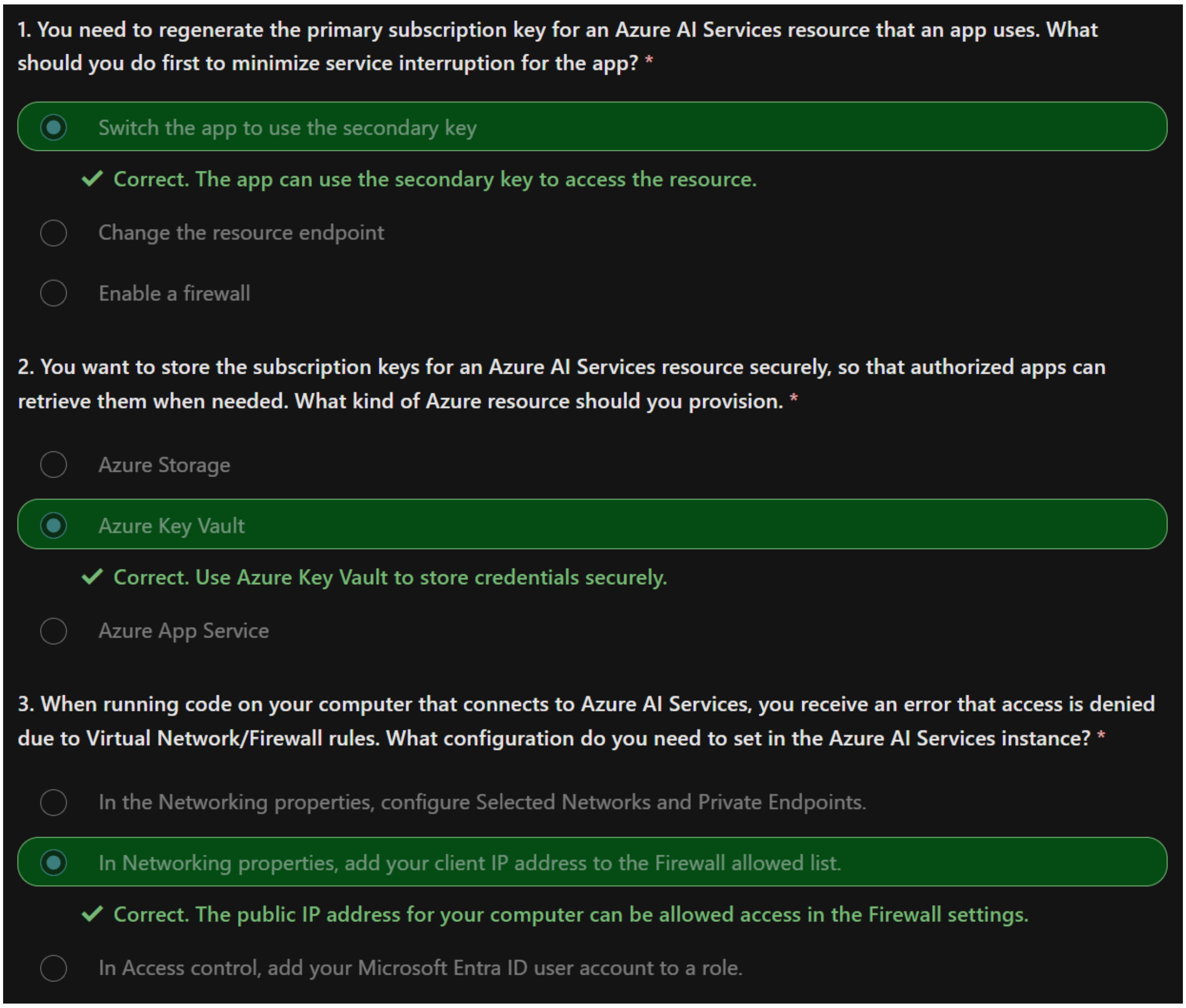
6. When prompted, enter some text and review the language that is detected by the service. For example, try entering "Hello", "Bonjour", and "Gracias".
7. When you have finished testing the application, enter "quit" to stop the program.

Clean up resources

If you're not using the Azure resources created in this lab for other training modules, you can delete them to avoid incurring further charges.

1. Open the Azure portal at `https://portal.azure.com`, and in the top search bar, search for the resources you created in this lab.
2. On the resource page, select **Delete** and follow the instructions to delete the resource. Alternatively, you can delete the entire resource group to clean up all resources at the same time.

Knowledge Check



1. You need to regenerate the primary subscription key for an Azure AI Services resource that an app uses. What should you do first to minimize service interruption for the app? *

[✓ Switch the app to use the secondary key]

✓ Correct. The app can use the secondary key to access the resource.

Change the resource endpoint

Enable a firewall

2. You want to store the subscription keys for an Azure AI Services resource securely, so that authorized apps can retrieve them when needed. What kind of Azure resource should you provision. *

Azure Storage

[✓ Azure Key Vault]

✓ Correct. Use Azure Key Vault to store credentials securely.

Azure App Service

3. When running code on your computer that connects to Azure AI Services, you receive an error that access is denied due to Virtual Network/Firewall rules. What configuration do you need to set in the Azure AI Services instance? *

In the Networking properties, configure Selected Networks and Private Endpoints.

[✓ In Networking properties, add your client IP address to the Firewall allowed list.]

✓ Correct. The public IP address for your computer can be allowed access in the Firewall settings.

In Access control, add your Microsoft Entra ID user account to a role.

Summary

In this module, you learned how to:

- Consider authentication for AI services.
- Manage network security for AI services.

For more information about securing AI services, see Azure AI services security in the service documentation.

✍️ Compiled by Kenneth Leung (2025)