1.2 - Create and consume Azure AI services

Overview

Azure AI Services is a collection of services that are building blocks of AI functionality you can integrate into your applications. In this learning path, you'll learn how to provision, secure, monitor, and deploy Azure AI Services resources and use them to build intelligent solutions.

Primer

- In Microsoft Azure, a **subscription** is a logical container that is used to provision and manage Azure resources. It provides a way to organize and manage access to Azure services and resources.
- A **resource** is a manageable item (aka instance) that is available through the Azure platform.
- A **resource group** in Microsoft Azure is a container that holds related resources for an Azure solution. It serves as a logical grouping that allows you to manage and organize resources such as virtual machines, storage accounts, virtual networks, databases, and web apps collectively.
- A **service** in Azure is a broader category of functionalities that you can use to perform tasks or run applications, such as computing, storage, networking, databases, and more.
- **Resource vs Service**: An Azure service is a broad offering that provides specific functionalities and capabilities, such as compute or storage, while a resource is an individual instance of an Azure service that you create and manage, such as a virtual machine or a storage account.

Introduction

Azure AI services are cloud-based services that encapsulate AI capabilities. Rather than a single product, you should think of AI services as a set of individual services that you can use as building blocks to compose sophisticated, intelligent applications.

AI services includes a wide range of individual services across multiple categories, as shown in the following table.

Language	Speech	Vision	Decision
Azure AI Language	Azure AI Speech	Azure AI Computer Vision	Azure AI Anomaly Detector
Azure AI Translator		Azure AI Custom Vision	Azure AI Content Moderator
		Azure AI Face	Azure AI Personalizer

You can use AI services to build your own AI solutions to provide out-of-the-box solutions for common AI scenarios. Azure AI services include:

- **Azure AI Document Intelligence** - An **optical character recognition (OCR)** solution that can extract semantic meaning from forms, such as invoices, receipts, and others.
- **Azure AI Immersive Reader** - A reading solution that supports people of all ages and abilities.
- **Azure AI Search** - A **cloud-scale search solution** that uses AI services to extract insights from data and documents.
- **Azure OpenAI** - An Azure Cognitive Service that provides access to the capabilities of OpenAI GPT-4.

While the details of each AI service can vary, the approach to provisioning and consuming them is generally the same.

In this module, you will learn how to:

- Create Azure AI services resources in an Azure subscription.
- Identify endpoints, keys, and locations required to consume an AI services resource.
- Use a REST API to consume an AI service.
- Use an SDK to consume an AI service.

Provision an Azure AI services resource

Azure AI services include a wide range of AI capabilities that you can use in your applications. To use any of the AI services, you need to create appropriate resources in an Azure subscription to define an endpoint where the service can be consumed, provide access keys for authenticated access, and to manage billing for your application's usage of the service.

Options for Azure resources

For many of the available AI services, you can choose between the following provisioning options:

Multi-service resource

You can provision an **AI services** resource that supports multiple different AI services. For example, you could create a **single resource** that enables you to use the **Azure AI Language**, **Azure AI Vision**, **Azure AI Speech**, and other services.

This approach enables you to manage a single set of access credentials to consume multiple services at a single endpoint, and with a single point of billing for usage of all services.

Single-service resource

Each AI service can be provisioned individually, for example by creating discrete **AI Language** and **AI Vision** resources in your Azure subscription.

This approach enables you to use **separate endpoints for each service** (for example to provision them in different geographical regions) and to **manage access credentials for each service independently**. It also enables you to manage billing separately for each service.

Single-service resources generally offer a free tier (with usage restrictions), making them a good choice to try out a service before using it in a production application.

Training and prediction resources

While most AI services can be used through a single Azure resource, some offer (or require) separate resources for model *training* and *prediction*. This enables you to manage billing for training custom models **separately** from model consumption by applications, and in most cases enables you to use a **dedicated service-specific resource to train a model**, but a generic **AI services** resource to make the model available to applications for inferencing.

Identify endpoints and keys

When you provision an Azure AI services service resource in your Azure subscription, you are defining an endpoint through which the service can be consumed by an application.

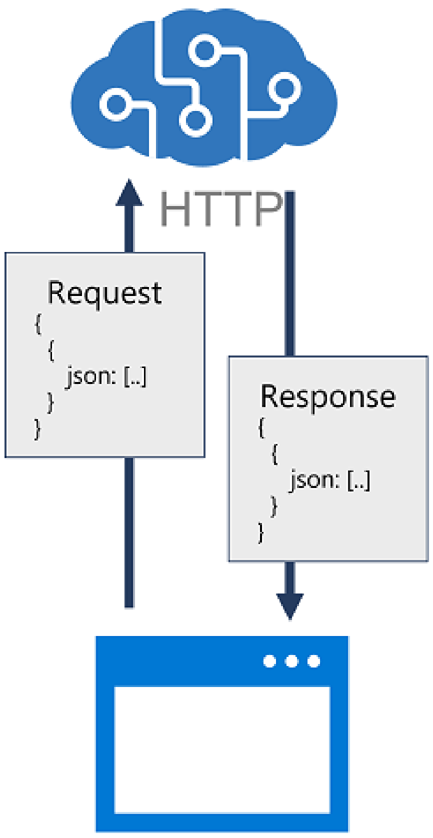
To consume the service through the endpoint, applications require the following information:

- **The endpoint URI**. This is the HTTP address at which the REST interface for the service can be accessed. Most AI services software development kits (SDKs) use the endpoint URI to initiate a connection to the endpoint.
- **A subscription key**. Access to the endpoint is restricted based on a subscription key. Client applications must provide a valid key to consume the service. **When you provision an AI services resource, two keys are created - applications can use either key.** You can also regenerate the keys as required to control access to your resource.
- **The resource location**. When you provision a resource in Azure, you generally assign it to a location, which **determines the Azure data center in which the resource is defined**. While most SDKs use the endpoint URI to connect to the service, some require the location.

Use a REST API

Azure AI services provide REST application programming interfaces (APIs) that client applications can use to consume services. In most cases, service functions can be called by **submitting data in JSON format over an HTTP request**, which may be a POST, PUT, or GET request depending on the specific function being called.

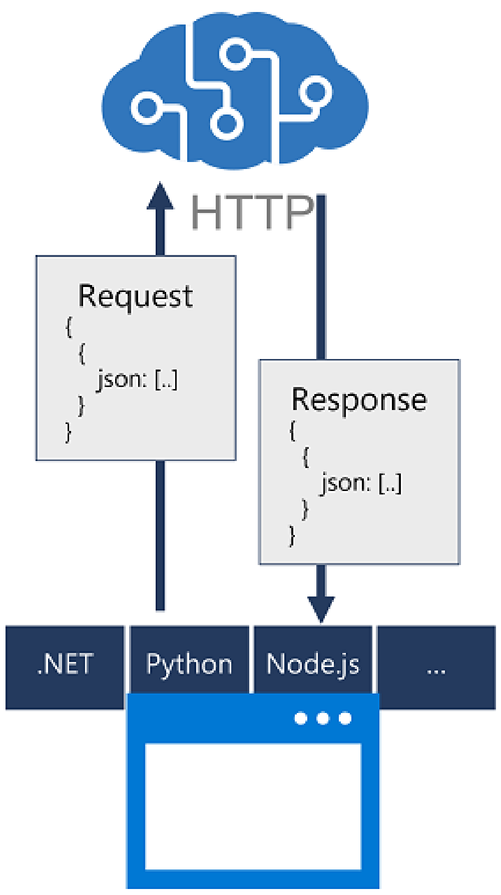
The **results of the function are returned to the client as an HTTP response, often with JSON contents that encapsulate the output data from the function**.



The use of REST interfaces with an HTTP endpoint means that any programming language or tool capable of submitting and receiving JSON over HTTP can be used to consume AI services. You can use common programming languages such as Microsoft C#, Python, and JavaScript; as well as utilities such as Postman and cURL, which can be useful for testing.

Use an SDK

You can develop an application that uses Azure AI services using REST interfaces, but it's **easier to build more complex solutions by using native libraries for the programming language** in which you're developing the application.



Software development kits (SDKs) for common programming languages abstract the REST interfaces for most AI services. SDK availability varies by individual AI services, but for most services there's an SDK for languages such as:

- Microsoft C# (.NET Core)
- Python
- JavaScript (Node.js)
- Go
- Java

Each SDK includes packages that you can install in order to use service-specific libraries in your code, and online documentation to help you determine the appropriate classes, methods, and parameters used to work with the service.

Exercise - Use Azure AI services

This unit includes a lab to complete. Use the free resources provided in the lab to complete the exercises in this unit. You will not be charged **for the lab environment**; however, you may need to bring your own subscription depending on the lab.

Microsoft provides this lab experience and related content for educational purposes. All presented information is owned by Microsoft and intended solely for learning about the covered products and services in this Microsoft Learn module.

To complete the exercise for this module, launch the VM and follow the instructions. To complete this exercise, you will need a Microsoft Azure subscription. If you don't already have one, you can sign up for one.

If you need to set up your computer for this exercise, you can use this setup guide and then follow the exercise instructions linked below. Note that the setup guide is designed for multiple development exercises, and may include software that is not required for this specific exercise. Additionally, due to the range of possible operating systems and setup configurations, we can't provide support if you choose to complete the exercise on your own computer.

In this exercise, you'll get started with Azure AI Services by creating an **Azure AI Services** resource in your Azure subscription and using it from a client application. The goal of the exercise is not to gain expertise in any particular service, but rather to become familiar with a general pattern for provisioning and working with Azure AI services as a developer.

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, 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/01-use-azure-ai-services` folder.

Provision an Azure AI Services resource

Azure AI Services are cloud-based services that encapsulate artificial intelligence capabilities you can incorporate into your applications. You can provision individual Azure AI services resources for specific APIs (for example, **Language** or **Vision**), or you can provision a single **Azure AI Services** resource that provides access to multiple Azure AI services APIs through a single endpoint and key. In this case, you'll use a single **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.
5. Go to the 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 key to authenticate).
    - The *location* where the resource is hosted. This is required for requests to some (but not all) APIs.

Use a REST Interface

> View the Python script here: rest-client.py

The Azure AI services APIs are REST-based, so you can consume them by submitting JSON requests over HTTP. In this example, you'll explore a console application that uses the **Language** REST API to perform language detection; but the basic principle is the same for all of the APIs supported by the Azure AI Services resource.

> **Note**: In this exercise, you can choose to use the REST API from either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

1. In Visual Studio Code, expand the **C-Sharp** or **Python** folder depending on your language preference.
2. View the contents of the **rest-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 **endpoint** and an authentication **key** for your Azure AI services resource. Save your changes.
3. Note that the **rest-client** folder contains a code file for the client application:

    - **C#**: Program.cs
    - **Python**: rest-client.py

    Open the code file and review the code it contains, noting the following details:
    - Various namespaces are imported to enable HTTP communication
    - Code in the **Main** function retrieves the endpoint and key for your Azure AI services resource - these will be used to send REST requests to the Text Analytics service.
    - The program accepts user input, and uses the **GetLanguage** function to call the Text Analytics language detection REST API for your Azure AI services endpoint to detect the language of the text that was entered.
    - The request sent to the API consists of a JSON object containing the input data - in this case, a collection of **document** objects, each of which has an **id** and **text**.
    - The key for your service is included in the request header to authenticate your client application.
    - The response from the service is a JSON object, which the client application can parse.
4. Right click on the **rest-client** folder, select *Open in Integrated Terminal* and run the following command:
    C#

```
dotnet run
```

Python

```
pip install python-dotenv
python rest-client.py
```

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

Use an SDK

> View the Python script here: sdk-client.py

You can write code that consumes Azure AI services REST APIs directly, but there are software development kits (SDKs) for many popular programming languages, including Microsoft C#, Python, Java, and Node.js. Using an SDK can greatly simplify development of applications that consume Azure AI services.

1. In Visual Studio Code, expand the **sdk-client** folder under the **C-Sharp** or **Python** folder, depending on your language preference. Then run `cd ../sdk-client` to change into the relevant **sdk-client** folder.
2. Install the Text Analytics SDK package by running the appropriate command for your language preference:
   C#

   ```
   dotnet add package Azure.AI.TextAnalytics --version 5.3.0
   ```

   Python

   ```
   pip install azure-ai-textanalytics==5.3.0
   ```

3. View the contents of the **sdk-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 **endpoint** and an authentication **key** for your Azure AI services resource. Save your changes.
4. Note that the **sdk-client** folder contains a code file for the client application:

   - **C#**: `Program.cs`
   - **Python**: `sdk-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 retrieves the endpoint and key for your Azure AI services resource - these will be used with the SDK to create a client for the Text Analytics service.

- 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. Return to the terminal, ensure you are in the **sdk-client** folder, and enter the following command to run the program:
   C#

   ```
   dotnet run
   ```

   Python

   ```
   python sdk-client.py
   ```

6. When prompted, enter some text and review the language that is detected by the service. For example, try entering "Goodbye", "Au revoir", and "Hasta la vista".
7. When you have finished testing the application, enter "quit" to stop the program.

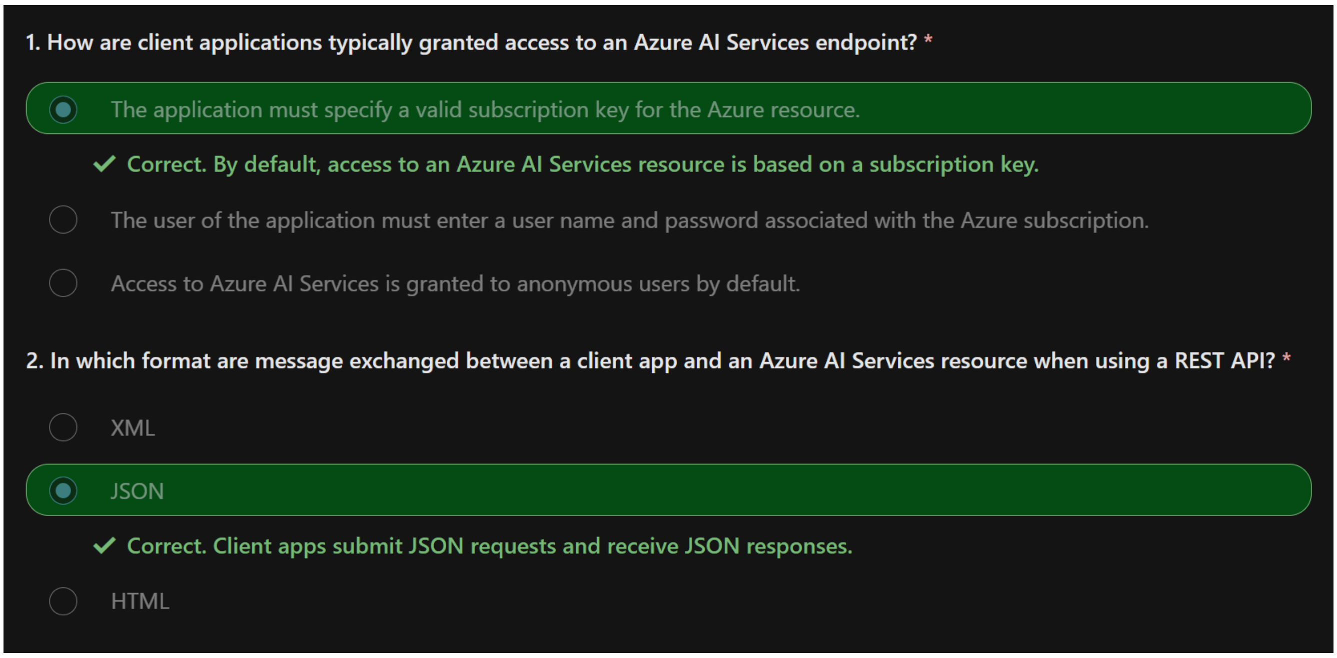
> **Note**: Some languages that require Unicode character sets may not be recognized in this simple console application.

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. How are client applications typically granted access to an Azure AI Services endpoint? *

[✓ The application must specify a valid subscription key for the Azure resource.]

✓ Correct. By default, access to an Azure AI Services resource is based on a subscription key.

○ The user of the application must enter a user name and password associated with the Azure subscription.

○ Access to Azure AI Services is granted to anonymous users by default.

2. In which format are message exchanged between a client app and an Azure AI Services resource when using a REST API? *

○ XML

[✓ JSON]

✓ Correct. Client apps submit JSON requests and receive JSON responses.

○ HTML

Summary

In this module, you learned how to:

- Create Azure AI services resources in an Azure subscription.
- Identify endpoints, keys, and locations required to consume an Azure AI service resource.
- Use a REST API to consume an Azure AI service.
- Use an SDK to consume an Azure AI service.

For more information about Azure AI services, refer to the AI services documentation.

✍️ Compiled by Kenneth Leung (2025)