1.5 - Deploy Azure AI services in containers

Overview

Learn about Container support in Azure AI services allowing the use of APIs available in Azure and enable flexibility in where to deploy and host the services with Docker containers.

After completing this module, learners will be able to:

- Create containers for reuse
- Deploy to a container and secure a container
- Consume Azure AI services from a container

Introduction

Containers enable you to host Azure AI services either on-premises or in Azure. For example, **if your application uses sensitive data in an on-premises SQL Server to call an Azure AI services service, you can deploy Azure AI services in containers on the same network.** Now your data can stay on your local network and not be passed to the cloud.

Deploying Azure AI services in a container on-premises will also decrease the latency between the service and your local data, which can improve performance.

In this module, you'll learn how to:

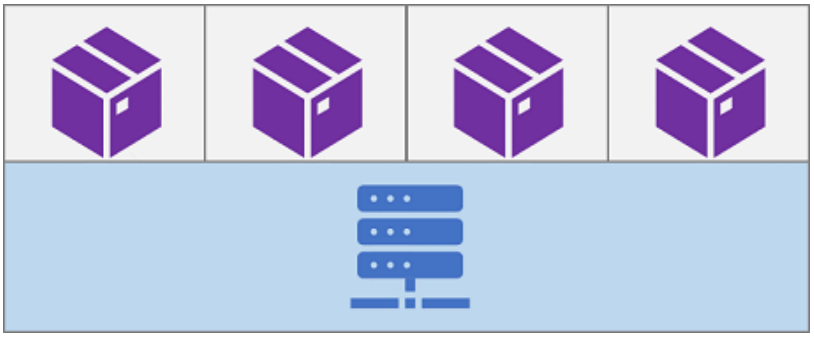
- Create containers for reuse.
- Deploy to a container and secure a container.
- Consume Azure AI services from a container.

Understand containers

When you deploy a software service, it must be hosted in an environment that provides the hardware, operating system, and supporting runtime components on which the service depends.

Azure AI services is provided as a cloud service, in which the service software is hosted in an Azure data center that provides the underlying runtime services, operating system, and hardware.

However, **you can also deploy some Azure AI services in a *container*, which encapsulates the necessary runtime components**, and which is in turn deployed in a container host that provides the underlying operating system and hardware.



What is a container?

A container comprises an application or service and the runtime components needed to run it, while abstracting the underlying operating system and hardware. In practice, this abstraction results in two significant benefits:

- Containers are portable across hosts, which may be running different operating systems or use different hardware - making it easier to move an application and all its dependencies.
- A single container host can support multiple isolated containers, each with its own specific runtime configuration - making it easier to consolidate multiple applications that have different configuration requirement.

A container is encapsulated in a *container image* that defines the software and configuration it must support. Images can be stored in a central registry, such as *Docker Hub*, or you can maintain a set of images in your own registry.

Container deployment

To use a container, you typically pull the container image from a registry and deploy it to a container host, specifying any required configuration settings. The container host can be in the cloud, in a private network, or on your local computer. For example:

- A *Docker** server.

- An Azure Container Instance (ACI).
- An Azure Kubernetes Service (AKS) cluster.

*Docker is an open source solution for container development and management that includes a server engine that you can use to host containers. There are versions of the Docker server for common operating systems, including Microsoft Windows and Linux.

Tip: To learn more about containers, review the Introduction to Docker containers module on Microsoft Learn.

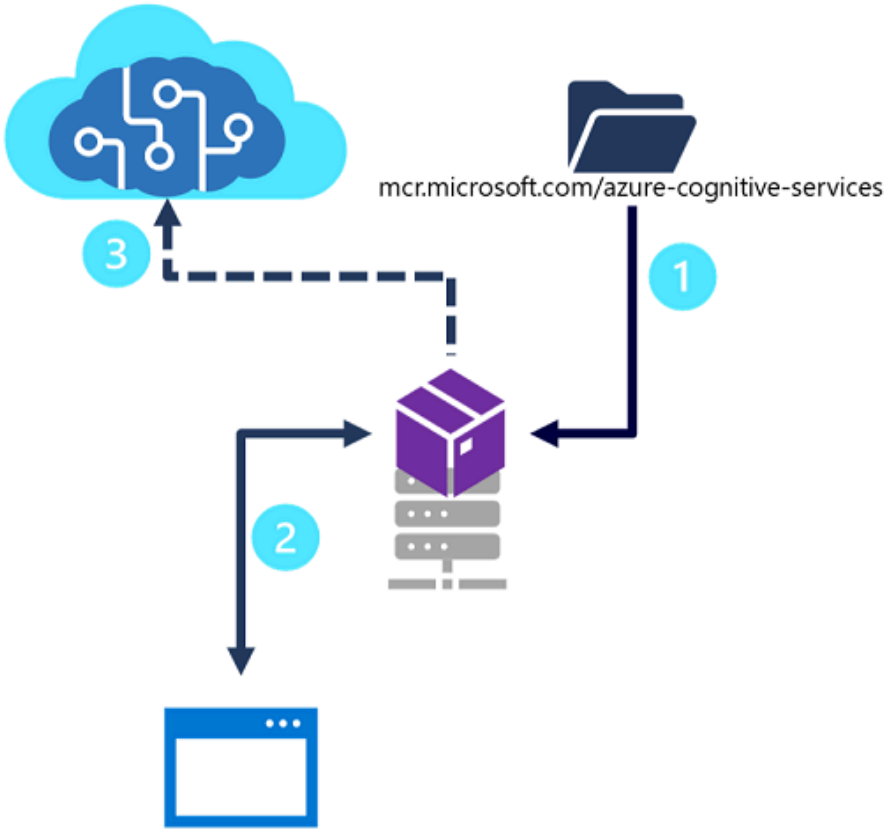
Use Azure AI services containers

There are container images for Azure AI services in the **Microsoft Container Registry** that you can use to **deploy a containerized service that encapsulates an individual Azure AI services service API**.

> Note (from ChatGPT): Typically, in the case of Azure AI services deployed via container images, the proprietary model weights are not included directly within the container image itself. Instead, these containers are designed primarily to facilitate interaction with Azure's cloud services. They act as a local gateway or interface for Azure AI services, managing API calls and processing user requests by communicating with the Azure cloud backend where the actual AI processing is done.
>
> The container image usually contains the necessary software and environment setup to make API calls to the Azure AI services. The containers handle API calls to Azure's cloud services, including request processing, authentication, and communication protocols. When you deploy an Azure AI container on your local machine or in your private cloud, it functions mainly to route requests to the Azure cloud, where the AI processing occurs using Microsoft's proprietary models.

To deploy and use an Azure AI services container, the following three activities must occur:

1. The container image for the specific Azure AI services API you want to use is downloaded and deployed to a container host, such as a local Docker server, an Azure Container Instance (ACI), or Azure Kubernetes Service (AKS).
2. Client applications submit data to the **endpoint provided by the containerized service**, and retrieve results just as they would from an Azure AI services cloud resource in Azure.
3. Periodically, usage metrics for the containerized service are sent to an Azure AI services resource in Azure in order to calculate billing for the service.



Even when using a container, **you must provision an Azure AI services resource in Azure** for billing purposes.

Client applications send their requests to the containerized service, meaning that potentially sensitive data is not sent to the Azure AI services endpoint in Azure; but the container must be able to connect to the Azure AI services resource in Azure periodically to send usage metrics for billing.

Azure AI services container images

Each container provides a subset of Azure AI services functionality. For example, not all features of the Azure AI Language service are in a single container. **Language detection, translation, and sentiment analysis are each separate container images.** However, the setup steps are similar for each container.

Language containers

For the AI Language service, the core features map to separate images:

Feature	Image
Key Phrase Extraction	mcr.microsoft.com/azure-cognitive-services/textanalytics/keyphrase
Language Detection	mcr.microsoft.com/azure-cognitive-services/textanalytics/language
Sentiment Analysis	mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment
Named Entity Recognition	mcr.microsoft.com/product/azure-cognitive-services/textanalytics/language/about
Text Analytics for health	mcr.microsoft.com/product/azure-cognitive-services/textanalytics/healthcare/about

Feature	Image
Translator	mcr.microsoft.com/product/azure-cognitive-services/translator/text-translation/about
Summarization	mcr.microsoft.com/azure-cognitive-services/textanalytics/summarization

> Note: Sentiment Analysis supports other languages by replacing the *en* in the image with the correct language code

Speech containers

Feature	Image
Speech to text	mcr.microsoft.com/product/azure-cognitive-services/speechservices/speech-to-text/about
Custom Speech to text	mcr.microsoft.com/product/azure-cognitive-services/speechservices/custom-speech-to-text/about
Neural Text to speech	mcr.microsoft.com/product/azure-cognitive-services/speechservices/neural-text-to-speech/about
Speech language detection	mcr.microsoft.com/product/azure-cognitive-services/speechservices/language-detection/about

Vision containers

Feature	Image
Read OCR	mcr.microsoft.com/product/azure-cognitive-services/vision/read/about
Spatial analysis	mcr.microsoft.com/product/azure-cognitive-services/vision/spatial-analysis/about

You can use the Docker *pull* command to download container images to work with them directly from your machine. Some of the containers are in a "Gated" public preview state, and you need to explicitly request access to use them. Otherwise the containers are available for anyone to use with their Azure AI services deployment.

For a full list of currently available Azure AI services container images, and specific notes for each one, see Azure AI services container image tags and release notes.

Azure AI services container configuration

When you deploy an Azure AI services container image to a host, you must specify three settings.

Setting	Description
ApiKey	Key from your deployed Azure AI service; used for billing.
Billing	Endpoint URI from your deployed Azure AI service; used for billing.
Eula	Value of **accept** to state you accept the license for the container.

Consuming Azure AI services from a Container

After your Azure AI services container is deployed, applications consume the containerized Azure AI services endpoint rather than the default Azure endpoint.

The client application must be configured with the appropriate endpoint for your container, but **does NOT need to provide a subscription key to be authenticated.** You can implement your own authentication solution and apply network security restrictions as appropriate for your specific application scenario.

Exercise

Use an Azure AI Services Container

Using Azure AI services hosted in Azure enables application developers to focus on the infrastructure for their own code while benefiting from scalable services that are managed by Microsoft. However, in many scenarios, organizations require more control over their service infrastructure and the **data that is passed between services.**

Many of the Azure AI services APIs can be packaged and deployed in a *container*, **enabling organizations to host Azure AI services in their own infrastructure; for example in local Docker servers, Azure Container Instances, or Azure Kubernetes Services clusters.**

Containerized Azure AI services need to communicate with an Azure-based Azure AI services account to support billing; but **application data is not passed to the back-end service**, and **organizations have greater control over the deployment configuration of their containers**, enabling custom solutions for authentication, scalability, and other considerations.

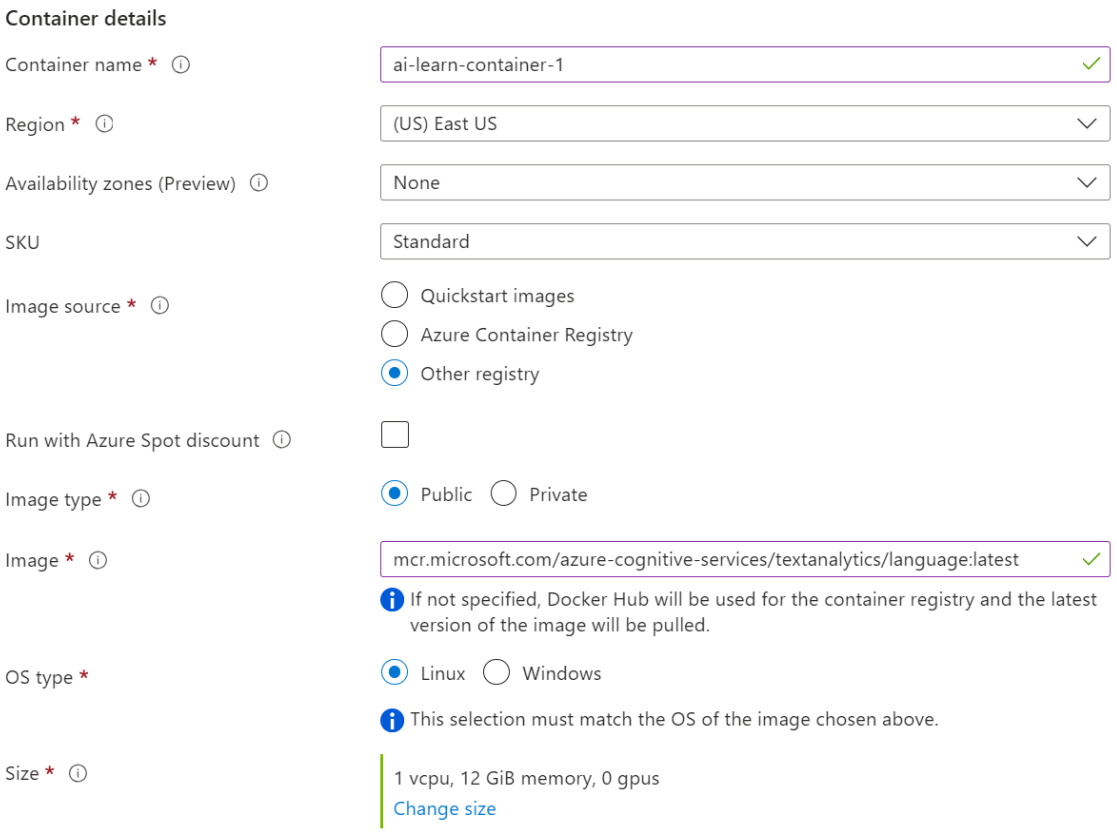
> **Note**: There is an issue currently being investigated that some users hit where containers won't deploy properly, and calls to those containers fail. Updates to this lab will be made as soon as the issue has been resolved.

Deploy and run a Text Analytics container

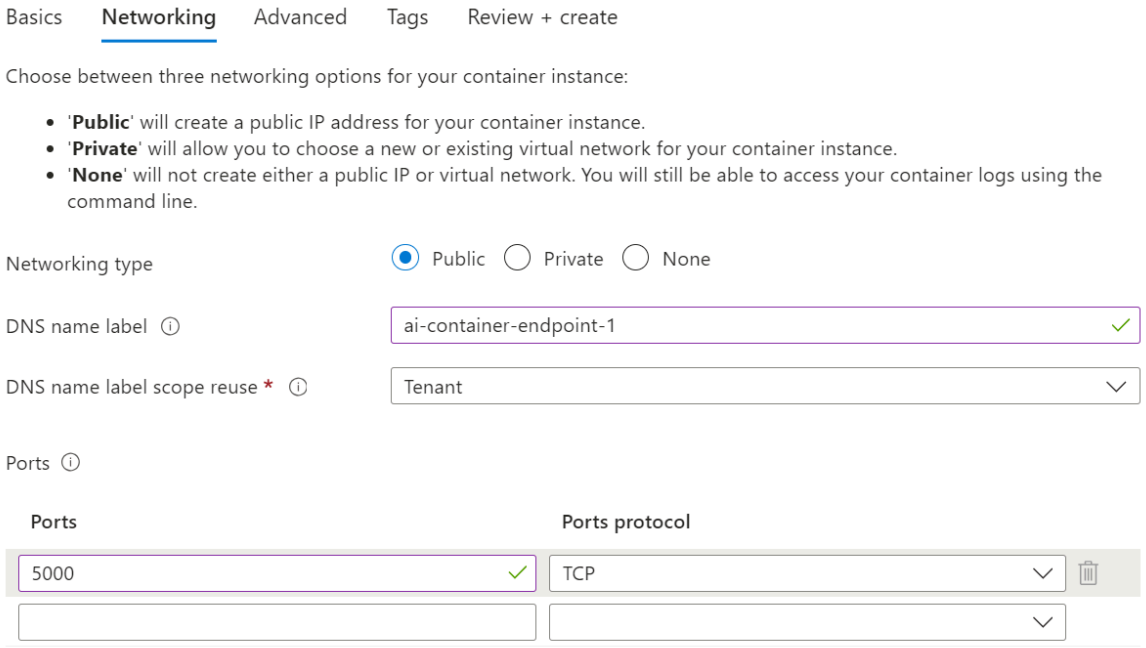
Many commonly used Azure AI services APIs are available in container images. For a full list, check out the Azure AI services documentation. In this exercise, **you'll use the container image for the Text Analytics *language detection* API; but the principles are the same for all of the available images.**

1. In the Azure portal, on the **Home** page, select the **+ Create a resource** button, search for *container instances*, and create a **Container Instances** resource with the following settings:
   - **Basics**:
     - **Subscription**: *Your Azure subscription*
     - **Resource group**: *Choose the resource group containing your Azure AI services resource*
     - **Container name**: *Enter a unique name*
     - **Region**: *Choose any available region*
     - **Image source**: Other Registry
     - **Image type**: Public
     - **Image**: `mcr.microsoft.com/azure-cognitive-services/textanalytics/language:latest`
     - **OS type**: Linux

- **Size**: 1 vcpu, 12 GB memory



- **Networking**:
  - **Networking type**: Public
  - **DNS name label**: *Enter a unique name for the container endpoint*
  - **Ports**: *Change the TCP port from 80 to **5000***

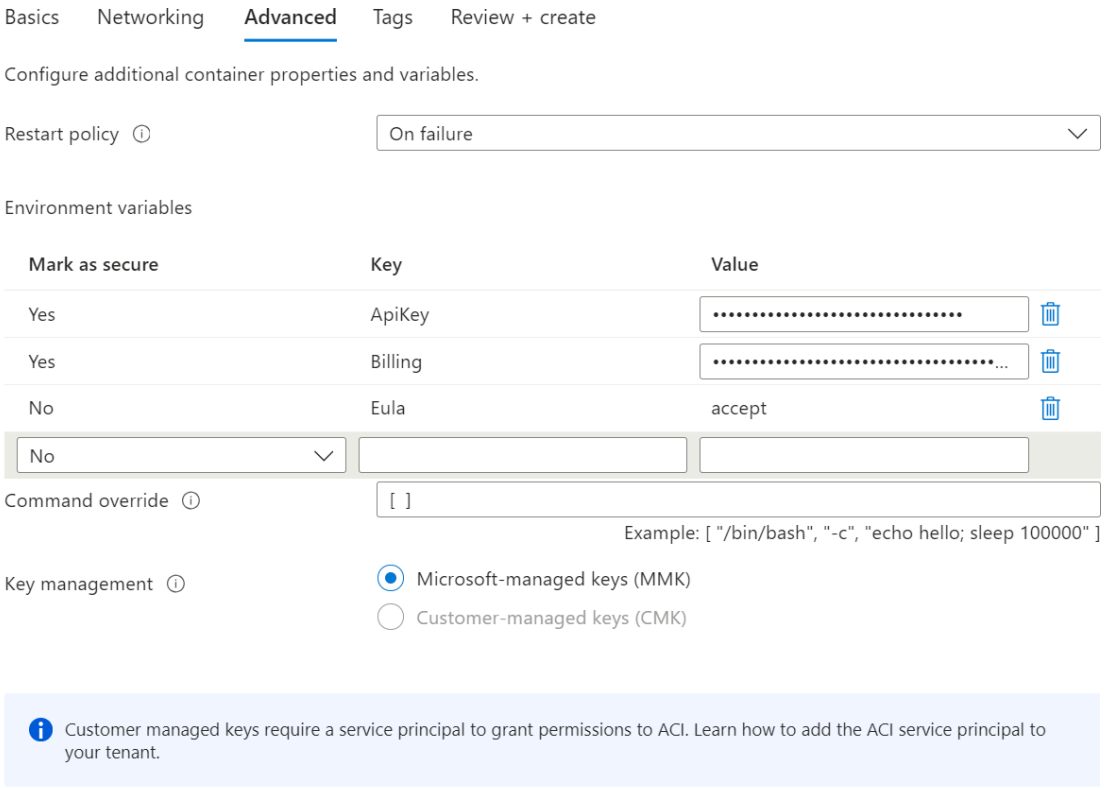


- **Advanced**:
  - **Restart policy**: On failure
  - **Environment variables**:
Mark as secure	Key	Value
Yes	`ApiKey`	*Either key for your Azure AI services resource*

| Yes | `Billing` | *The endpoint URI for your Azure AI services resource* |
| No | `Eula` | accept |

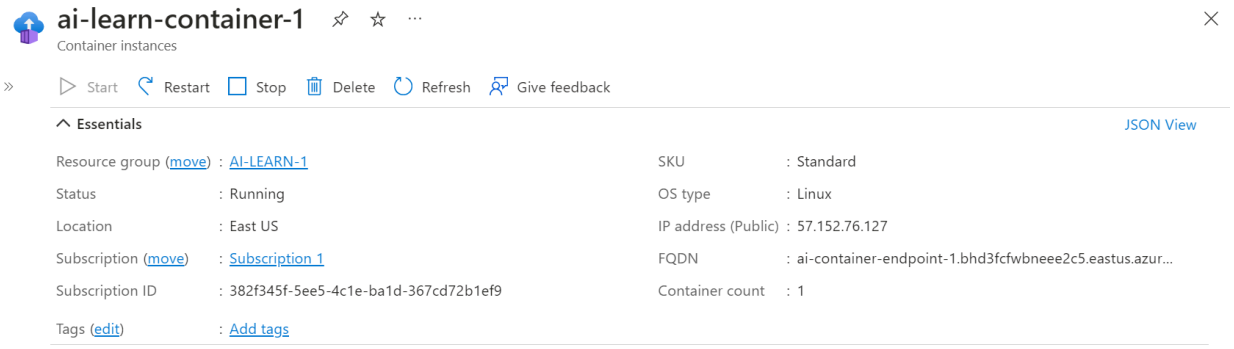
- *Example of endpoint URI:* `https://ai-learn-kl-1.cognitiveservices.azure.com`

- **Command override**: [ ]



- **Tags**:
  - *Don't add any tags*

2. Select **Review + create** then select **Create**. Wait for deployment to complete, and then go to the deployed resource.
3. **Note**: Please note that deploying an Azure AI container to **Azure Container Instances** typically takes 5-10 minutes (provisioning) before they are ready to use.
4. Observe the following properties of your container instance resource on its **Overview** page:
   - **Status**: This should be *Running*.
   - **IP Address**: This is the **public IP address you can use to access your container instances**.
   - **FQDN**: This is the *fully-qualified domain name* of the container instances resource, you can use this to access the container instances instead of the IP address.



> **Note**: In this exercise, you've deployed the Azure AI services container image for text translation to an **Azure Container Instances (ACI) resource**. You can use a similar approach to deploy it to a *Docker* host on your own computer or network by running the following command (on a single line) to deploy the language detection container to your local Docker instance,

> replacing `<yourEndpoint>` and `<yourKey>` with your endpoint URI and either of the keys for your Azure AI services resource. The command will look for the image on your local machine, and if it doesn't find it there it will pull it from the *mcr.microsoft.com* image registry and deploy it to your Docker instance. **When deployment is complete, the container will start and listen for incoming requests on port 5000.**

```
docker run --rm -it -p 5000:5000 --memory 12g --cpus 1 mcr.microsoft.com/azure-cognitive-services/textanalytics/language:latest Eula=accept Billing=<yourEndpoint> ApiKey=<yourKey>
```

Use the container

1. In your editor, open **rest-test.cmd** and edit the **curl** command it contains (shown below), replacing `<your_ACI_IP_address_or_FQDN>` with the IP address or FQDN for your container.

```
curl -X POST
"http://<your_ACI_IP_address_or_FQDN>:5000/text/analytics/v3.0/languages" -H
"Content-Type: application/json" --data-ascii "{'documents'
[{'id':1,'text':'Hello world.'},{'id':2,'text':'Salut tout le monde.'}]}"
```

2. Save your changes to the script by pressing **CTRL+S**.
3. **Note that you do not need to specify the Azure AI services endpoint or key - the request is processed by the containerized service**. The container in turn communicates periodically with the service in Azure to report usage for billing, but does not send request data.
4. Enter the following command to run the script:

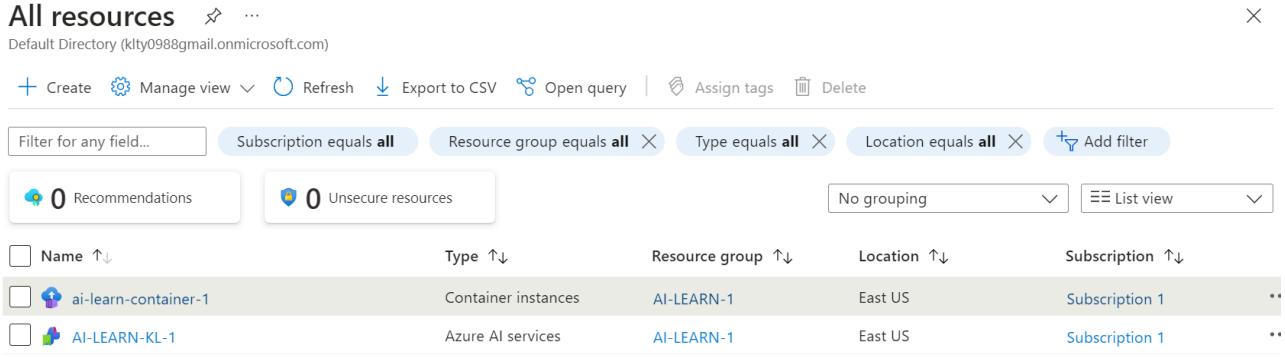
```
./rest-test.cmd
```

5. Verify that the command returns a JSON document containing information about the language detected in the two input documents (which should be English and French).

Clean Up

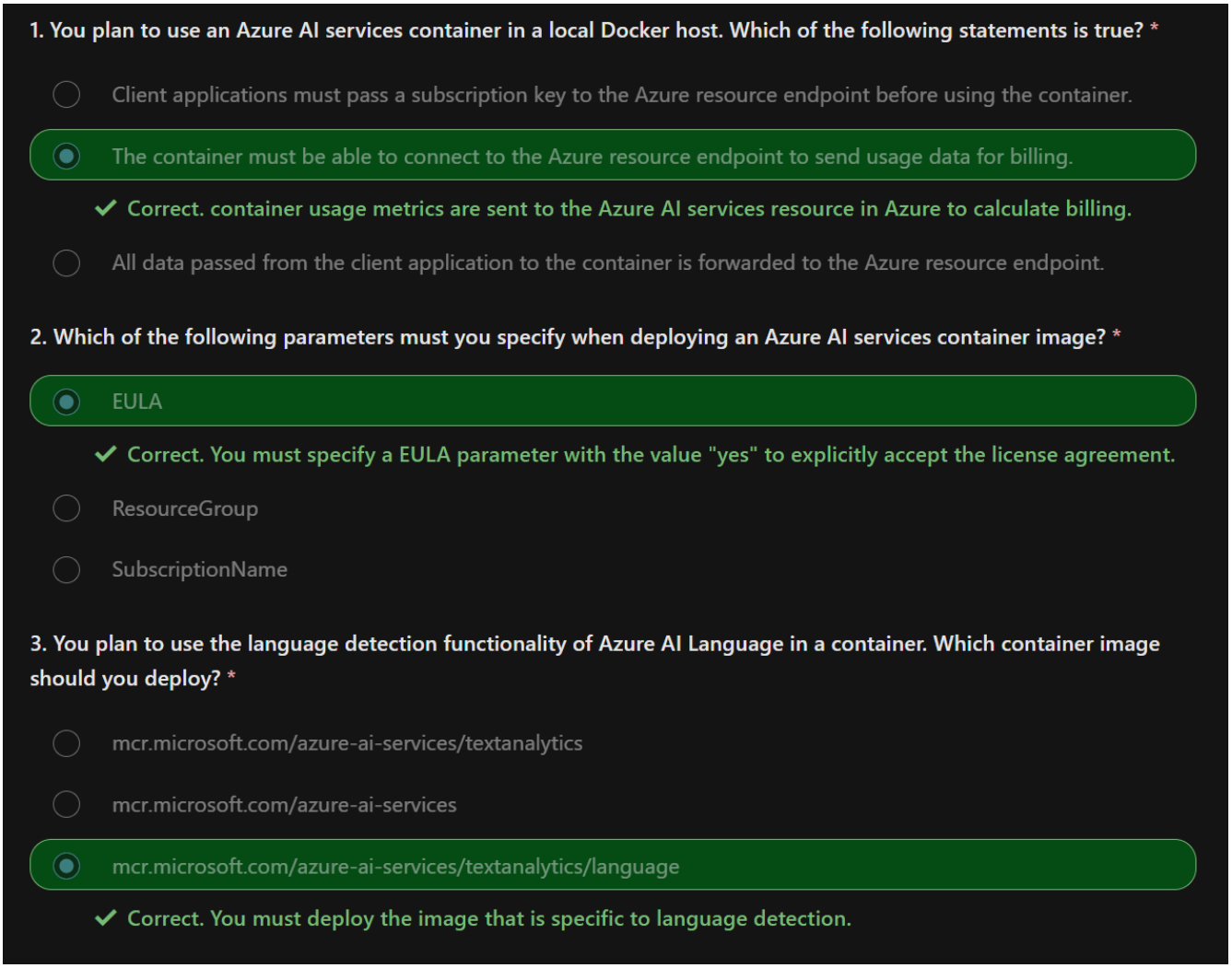
If you've finished experimenting with your container instance, you should delete it.

1. In the Azure portal, open the resource group where you created your resources for this exercise.



2. Select the container instance resource and delete it.

Knowledge Check



Summary

In this module, you learned how to:

- Create containers for reuse.
- Deploy to a container and secure a container.
- Consume Azure AI services from a container.

For more information about AI services containers, see Azure AI services containers in the AI services documentation.

✍️ Compiled by Kenneth Leung (2025)