

Detecting Faces with the Azure AI Vision



Learning Objectives

After completing this module, you will be able to:

- 1 Understand features, use cases, and responsibility of the Azure AI Vision Face API
- 2 Use the Face API in an app

Options for Face Detection, Analysis, and Recognition

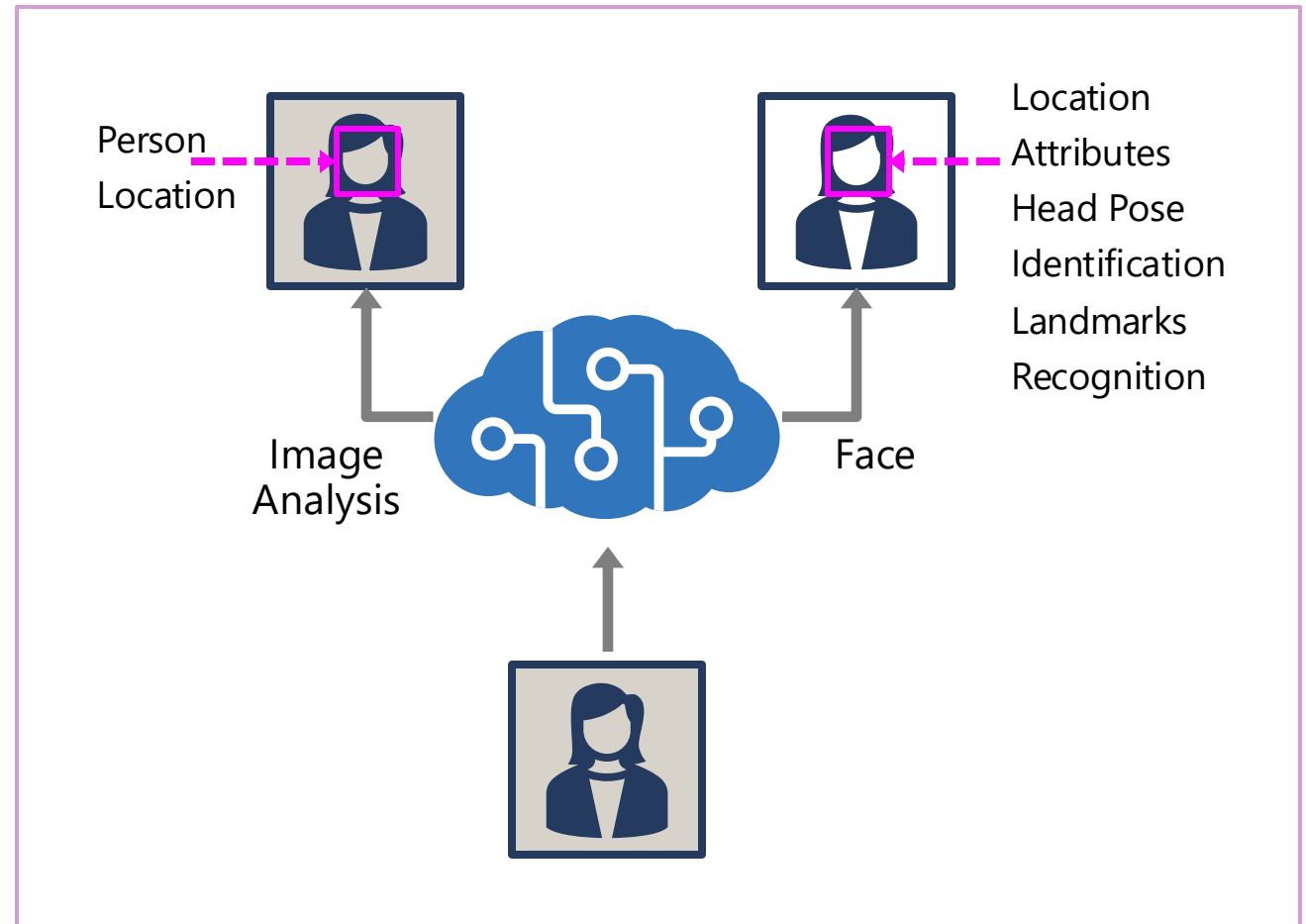
Image Analysis

- People detection
- Only location provided

Face Service

- Face detection
- Comprehensive facial feature analysis
- Face comparison and identification*
- Facial recognition*

* *Require Limited Access approval*



Considerations for Face Detection and Facial Recognition

Principles of responsible AI apply to all kinds of application, but systems that rely on facial data can be particularly problematic. As a safeguard for responsible AI usage, facial recognition, identification, verification, and comparison is behind a Limited Access policy, requiring users to be approved by Microsoft before enabling these features.

Data privacy and security

Systems based on facial data should protect individual privacy, ensuring that personally identifiable data is not accessed inappropriately

Transparency

Users should be informed about how their image will be used, and who will have access to it.

Fairness and Inclusiveness

Facial recognition should not be used in a manner that is prejudicial to individuals based on their appearance, or to unfairly target individuals

The Face Service

Face detection

Face attribute analysis

Facial landmark location

- Nose, eyes, mouth, ...

Face comparison*

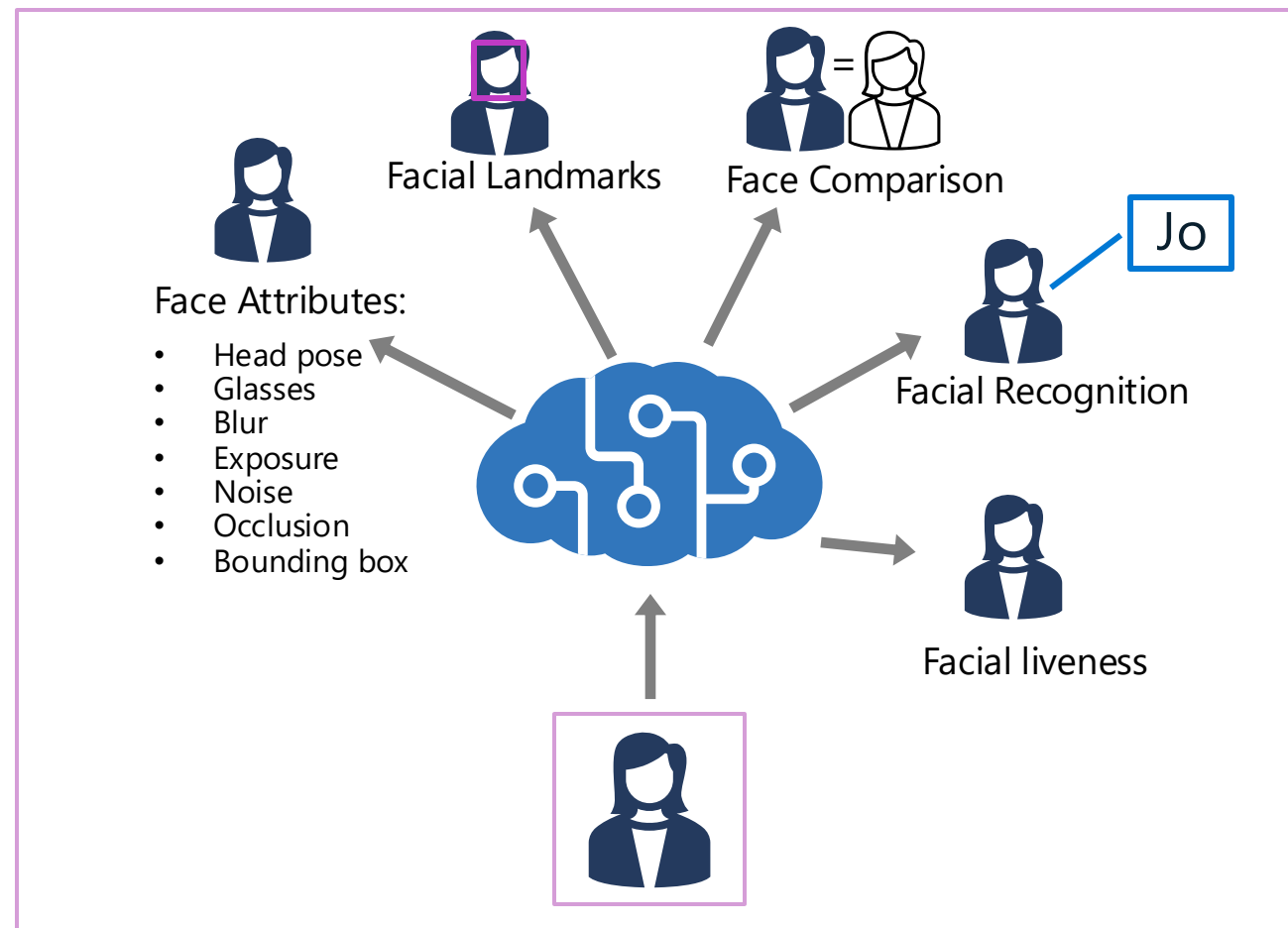
Facial recognition and identification*

Facial liveness*

Can be used as:

- Standalone **Face** resource
- Multi-service **Azure AI Services** resource

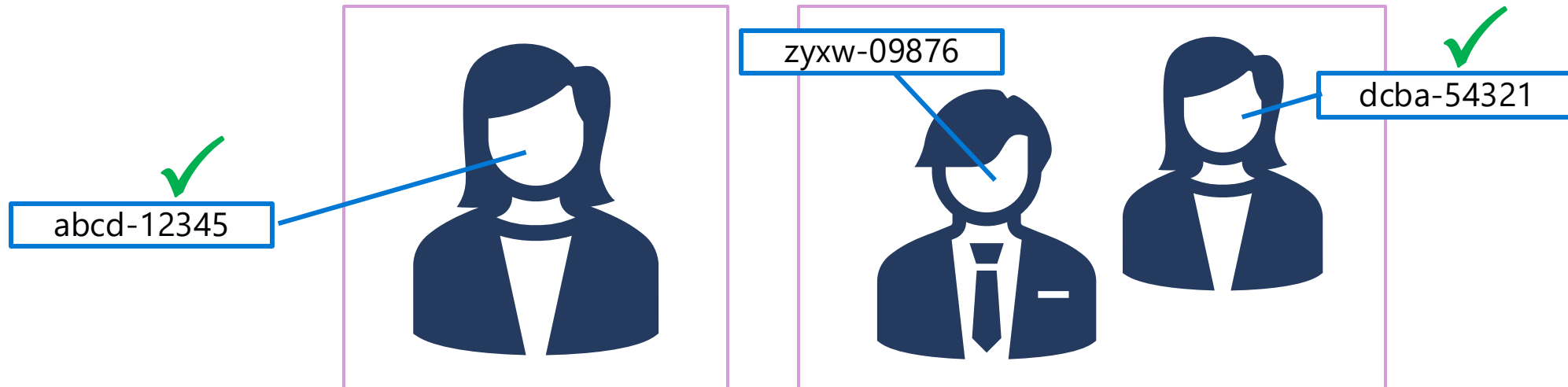
* *Require Limited Access approval*



Detected Face Identification

Every *detected* face is assigned an anonymous ID

- Retained in your service resource for 24 hours
- Can be used to compare faces in multiple images
 - *Verify* faces to determine if they're the same individual
 - *Find similar* faces to identify faces with similar features



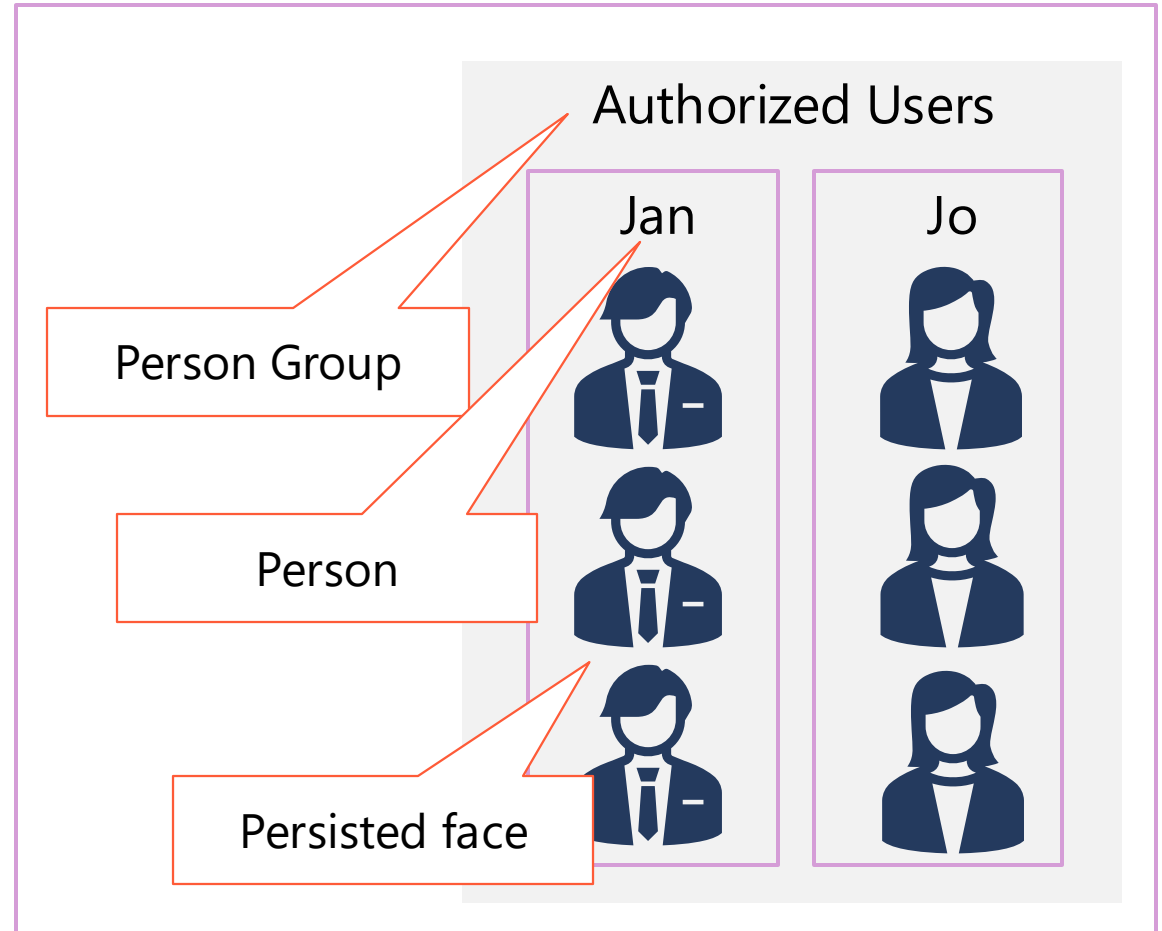
Persisted Face Recognition

Train a facial recognition model using face images

1. Create a **Person Group** for the people you want to identify
2. Add a **Person** for each individual
3. Add multiple detected **Faces** to each person
 - These become *persisted* faces
4. Train the model

Use the model for facial recognition

- *Identify* an individual person
- *Verify* the face of an individual person
- *Find* similar faces to a persisted face



Face Detection with Azure AI Vision

Use the **Face** endpoint, specifying **Faces** as a visual feature

Note: Most features like recognition and identification are not enabled for new users

Optional request parameters

returnFaceId	recognitionModel
returnFaceLandmarks	returnRecognitionModel
returnFaceAttributes	detectionModel



Request: `https://{endpoint}/face/v1.0/detect[?options]`
Body: `{"url": "http://path-to-image"}`

Response:

```
[
  {
    "faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
    "recognitionModel": "recognition_03",
    "faceRectangle": {
      "width": 78,
      "height": 78,
      "left": 394,
      "top": 54
    },
    "faceLandmarks": {
      "pupilLeft": {
        "x": 412.7,
        "y": 78.4
      },
      "pupilRight": {
        "x": 446.8,
        "y": 74.2
      }
    },
    ...
  }
]
```


Demo – Detect and Analyze Faces



Detect faces with the Azure AI Vision service

Detect and analyze faces with the Face service

Develop solutions with Azure AI Document Intelligence



Agenda



- Use prebuilt Document Intelligence models
- Train a custom Document Intelligence model

Develop a Document Intelligence solution



Learning Objectives

After completing this module, you will be able to:

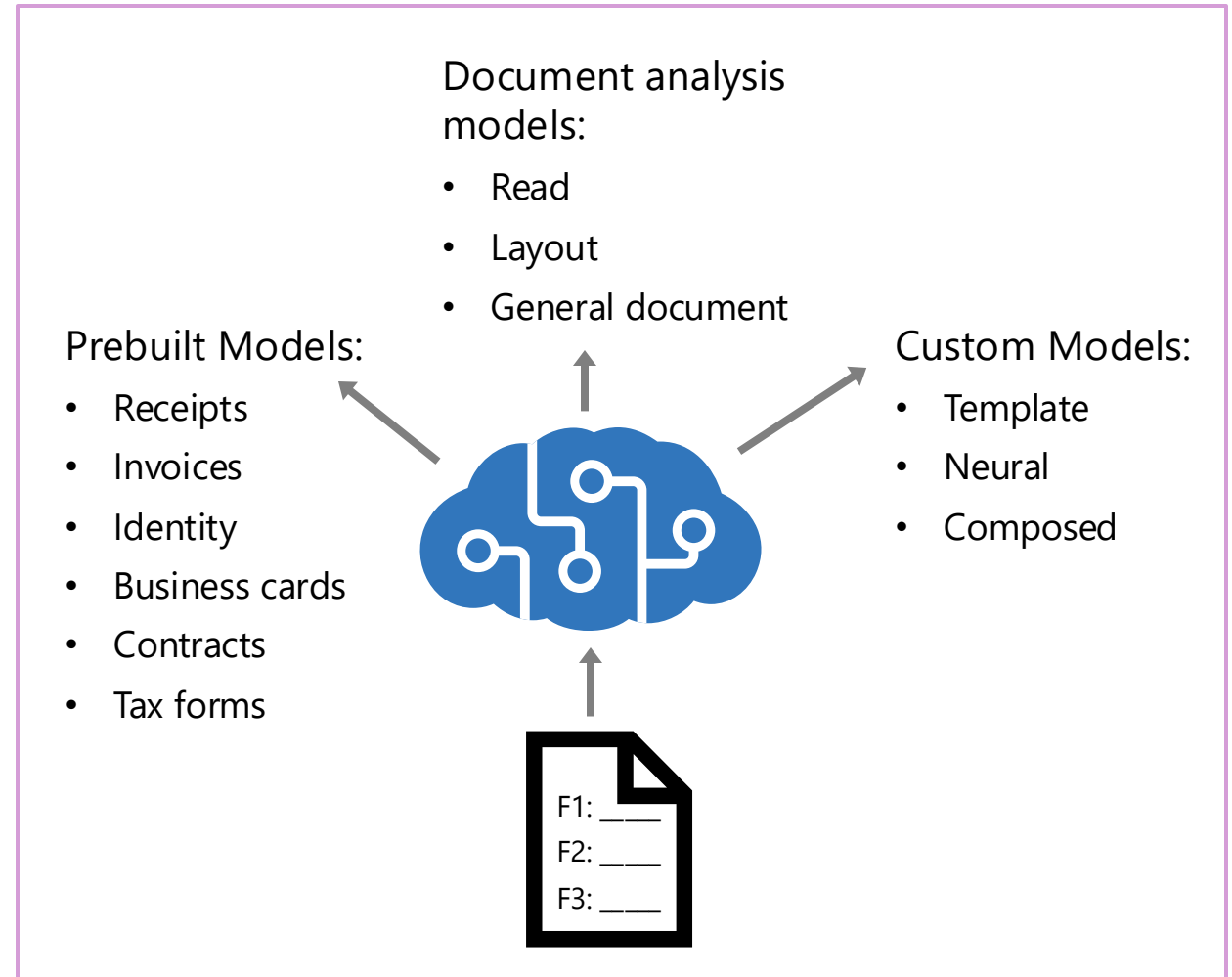
- 1 Understand models in Azure AI Document Intelligence
- 2 Train a custom Document Intelligence model
- 3 Connect an app to Document Intelligence APIs

The Document Intelligence Service

Data extraction from forms and documents:

- Document analysis from general documents
 - Read: OCR for printed and written text
 - Layout: Extract text and structure
 - General document: Extract text, structure, and key-value pairs
- Prebuilt models for common form types
- Train custom models for your own forms
 - Custom template: Extract data from static layouts
 - Custom neural: Extract data from mixed-type documents
 - Custom composed: Collection of multiple models assigned to a single model

Provision as single-service **Document Intelligence** resource or multi-service **Azure AI Services** resource



Prebuilt models

Receipt

Fourth Coffee		
1/1/2021: 09:34		
Latte	1	\$1.29
Cake	1	\$2.46

Total:		\$3.75

Analyze Receipt

Get Analyze Receipt Result

```
{
  "MerchantName": "Fourth Coffee",
  "TransactionDate": 2021-01-01,
  "TransactionTime": 09:34,
  "Items" [
    ...],
  "Total": 3.75
}
```

Invoice

Contoso		
Invoice No: 1234		
Date: 1/1/2021		
Item	Qty	Unit
X12	1	1.00
B38	1	2.99
Total: 3.99		

Analyze Invoice

Get Analyze Invoice Result

```
{
  "VendorName": "Contoso",
  "InvoiceNumber": 1234,
  "InvoiceDate": 2021-01-01
  "Tables" [
    ...],
  "TotalInvoiceAmount": 3.99
}
```

Business Card

Fabricam	
Hank Zoeng	
Sales director	
hank@fabrikam.com	
555-123-4567	

Analyze Business Card

Get Analyze Business Card Result

```
{
  "ContactNames": [
    {
      "FirstName": "Hank",
      "LastName": "Zoeng"
    }
  ],
  ...
}
```

Calling the API

- Each request is configured with your resource endpoint and needs your resource key
- Send the request, which when successful returns a poller to get the results
 - REST returns it in Operation-Location header
 - SDKs return an object from the request
- Query the poller received for the extracted data

REST

Request POST:

```
{endpoint}/documentintelligence/documentModels/prebuilt-layout:analyze?api-version={version}
```

Operation-Location:

```
{endpoint}/documentintelligence/documentModels/prebuilt-layout/analyzeResults/ab12345c-12ab-23cd-b19c-2322a7f11034?api-version={version}
```

C#

```
AnalyzeDocumentOperation operation = await  
client.AnalyzeDocumentFromUriAsync(WaitUntil.Completed,  
"prebuilt-layout", fileUri);
```

```
AnalyzeResult result = operation.Value;
```

Python

```
poller=document_analysis_client.begin_analyze_document_  
from_url("prebuilt-document", docUrl)
```

```
result = poller.result()
```


API response

- Response is broken down by page, lines, and words
- Subset of REST response included here
- SDK response objects have similar structure, broken down similarly
- Additional data about detected text or selection marks, such as bounding box and handwritten style

```
{
  "analyzeResult": {
    "apiVersion": "{version}",
    "modelId": "prebuilt-invoice",
    ...
    "pages": [{
      "pageNumber": 1,
      "angle": 0,
      "width": 8.5,
      "height": 11,
      "unit": "inch",
      "words": [{
        "content": "Margie's",
        "boundingBox": [
          0.5911,
          0.6857,
          1.7451,
          0.6857,
          1.7451,
          ...
        ],
        "confidence": 1,
        "span": {...}
      }],
    }],
  }
}
```

Demo – Use prebuilt Document Intelligence models



Use the Read model

Use an app to use document analysis

Types of custom models

Custom classification

- Apply a label to the entire document
- Ideal for sorting large numbers of incoming documents into types
- Requires two different classes, and a minimum of five labeled documents per class
- One type of training model

Custom extraction

- Apply label to specific text
- Ideal for extracting custom labels from documents
- Requires five examples of the same document type
- Two training methods:
 - **Custom template (custom form)**
 - Training time: 1-5 minutes
 - Document structure: forms, templates, other structured documents
 - **Custom neural (custom document)**
 - Training time: 20-60 minutes
 - Document structure: structured and unstructured documents

Training Custom Models

- 1 Create project and upload training files to your project, or connect to blob storage containing files
- 2 Add data type (such as field or signature) to start labeling your dataset
- 3 Select a word in the document, and assign one of the fields to label it
- 4 Repeat for all fields and files in your dataset
- 5 Layout and auto label (using a prebuilt model) can assist in this process
- 6 Train the model, providing a Model ID used in API requests

Document Intelligence Studio > Custom extraction model > customextract > Label data

Label data

Train

Drag & drop file here or [Browse for files](#)

Form_5.jpg
Form_4.jpg
Form_2.jpg
Form_3.jpg
Form_1.jpg

Run layout | Auto label | Draw region | Search | View | Add a field

1 Additional details

Bozeman MT 83839 Phone: 938-294-2949

Shipped From
Name: Wesley Smith
Company Name: We Sew
Address: 998 N Groove Road
Seattle WA 83838 Phone: 334-244-2949

Details	Quantity	Unit Price	Total
Black Sweats	20	10.00	
Black Yoga Pants	20	10.00	
White Sweats	20	10.00	
Yellow T Shirts	20	10.00	
Lean Stinkare	20	5.00	

Search existing or create new

Additional details

Wesley Smith
Wesley Smith
Manager

SUBTOTAL	\$
TAX	\$
TOTAL	\$1

Additional Notes:
We love to Print! Contact us about special offers on personalizing your future orders with company logos, cool designs, signatures, or pictures! We can put anything on clothing look forward to you being a return customer!

< 1 of 1 > Search View Print Refresh

Accuracy and confidence scores

- After training, a custom model has an estimated accuracy score
- Score is calculated by running combinations of training data predictions against the labeled values
- Confidence score is the same as using prebuilt models, indicated how accurate the model thinks that specific prediction is
- Confidence scores are provided in the response from the model for each predicted label

Accuracy

Email	80.00 %
CompanyAddress	80.00 %
Signature	80.00 %

Confidence

<div><div></div>Signature #1</div>	44.80%
Wesley Smith	
<div><div></div>CompanyAddress #1</div>	66.70%
342 W Wrinkle Road Bozeman MT 83839	
<div><div></div>Email #1</div>	95.30%
accounts@herolimited.com	

Analyze document using custom model

- Requires endpoint and key from deployed resource, similar to prebuilt models
- Needs to also include the ID of your deployed custom model
- Query the poller received for the extracted data

C#

```
AnalyzeDocumentOperation operation = await  
client.AnalyzeDocumentFromUriAsync(WaitUntil.Completed,  
{modelId}, {fileUri});
```

```
AnalyzeResult result = operation.Value;
```

Python

```
poller = document_analysis_client.begin_analyze_document(  
    model_id={modelId}, document={fileUri}  
  
result = poller.result()
```

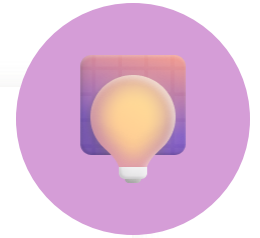
Lab – Extract Data from Forms



Train a custom model in Document Intelligence Studio

Test your custom Document Intelligence model

Knowledge check



1 You have scanned a letter into PDF format and need to extract the text it contains. What should you do?

- ☐ Use the Image Analysis feature in Azure AI Vision.
- ☒ Use the Read model in the Document Intelligence service.
- ☐ Use a custom model in the Document Intelligence service.

2 You need to build an application that submits expense claims, extracting the merchant, date, and amount from scanned receipts. What's the best way to do this?

- ☐ Use the general document model.
- ☐ Use the prebuilt Contract model.
- ☒ Use the prebuilt Receipt model.

3 You need to extract only data from specific fields in cargo manifest forms using Document Intelligence. What should you do?

- ☐ Use a prebuilt model.
- ☐ Build a custom composed model from several custom models.
- ☒ Train custom template model with labeled documents.

Creating a Knowledge Mining Solution



Agenda



- Implementing an Intelligent Search Solution
- Developing Custom Skills for an Enrichment Pipeline
- Creating a Knowledge Store

Implementing an Intelligent Search Solution



Learning Objectives

After completing this module, you will be able to:

- 1 Create an Azure AI Search Solution
- 2 Implement a custom skill for Azure AI Search and integrate it into a skillset
- 3 Create a knowledge store with object, file, and table projections

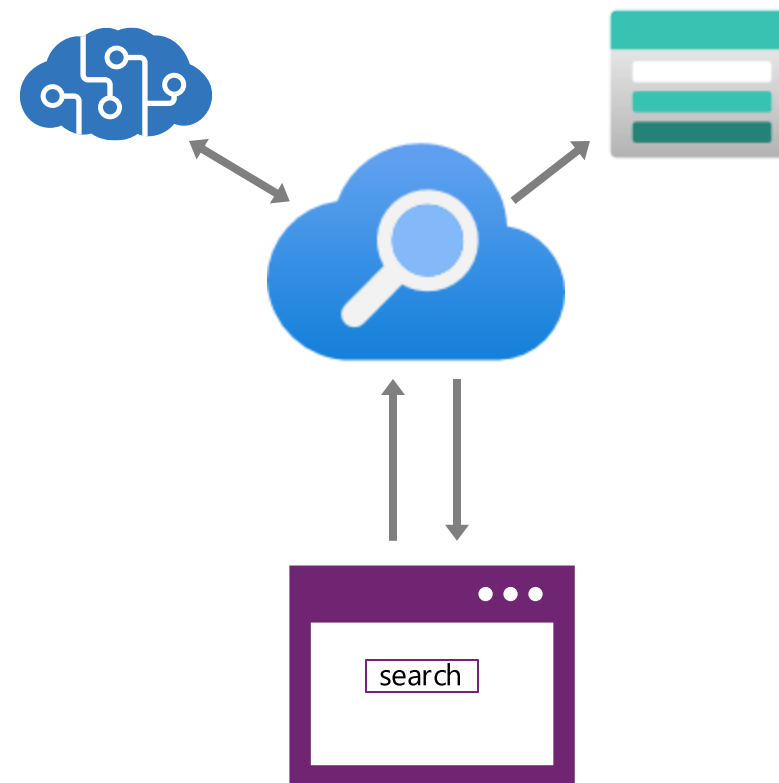
Azure AI Search

AI-Powered Knowledge Mining

- Index documents and data from a range of sources
- Use skills to enrich index data
- Store extracted insights in a knowledge store for analysis and integration

Azure Resources:

- **Azure AI Search** for core indexing and querying
- **Azure AI Services** for index enrichment
- **Storage account** for knowledge store persistence



Core Components of a AI Search Solution



Data Source

The data store to be searched:

- Blob storage container
- SQL Database
- Cosmos DB

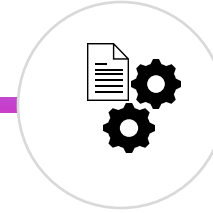
You can also push JSON documents directly into an index



Skillset

Defines an enrichment pipeline of AI skills to enhance data during indexing:

- Built-in AI skills
- Custom skills



Indexer

Maps data source fields and skillset outputs to index fields

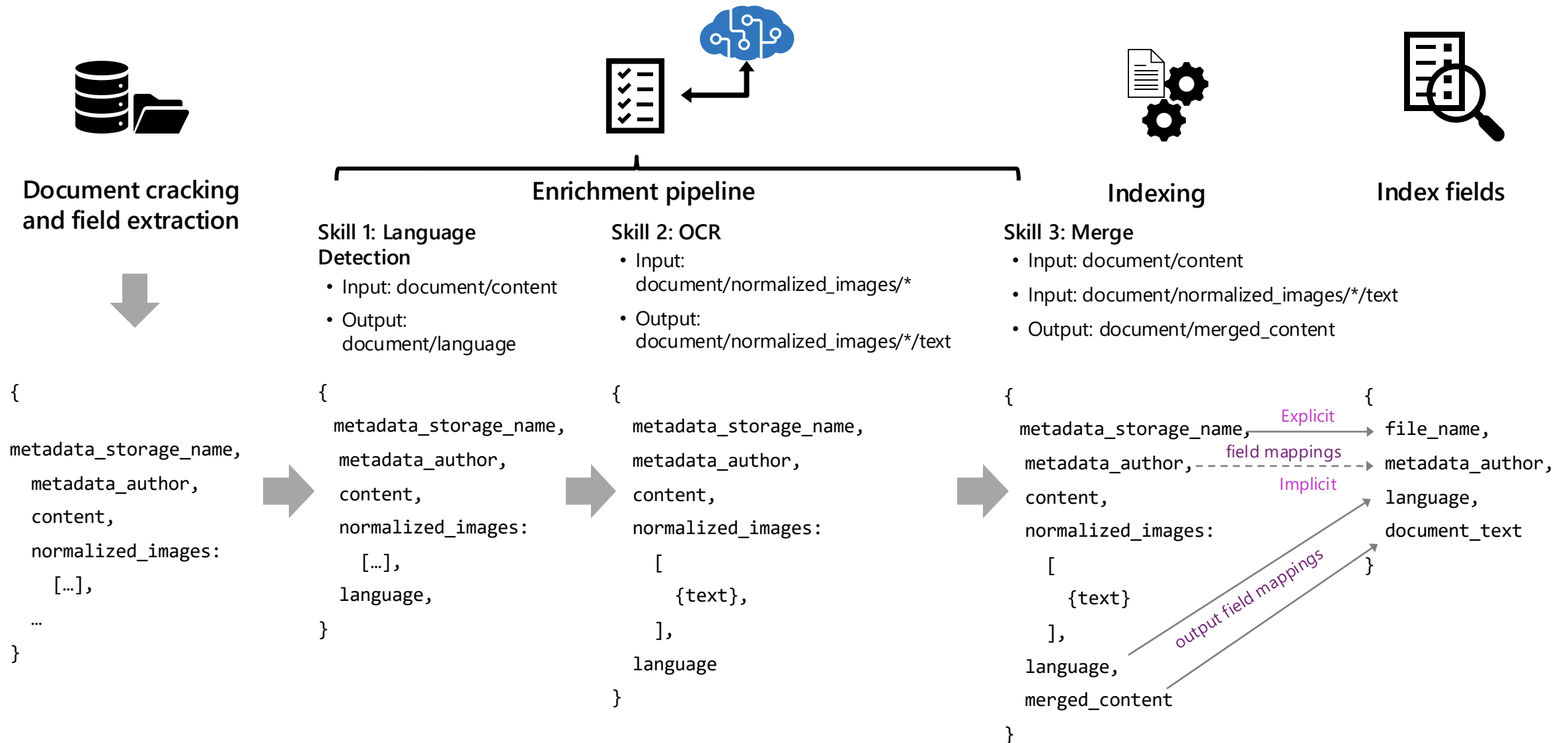
- Running the indexer builds the index



Index

Searchable collection of JSON documents containing extracted and enriched fields

How an Enrichment Pipeline Works



Demo – Create an Azure Cognitive Search Solution



Create an indexing solution

Modify an indexing solution

Query an index from a client application

Create a custom skill for Azure AI Search



Introduction to Custom Skills

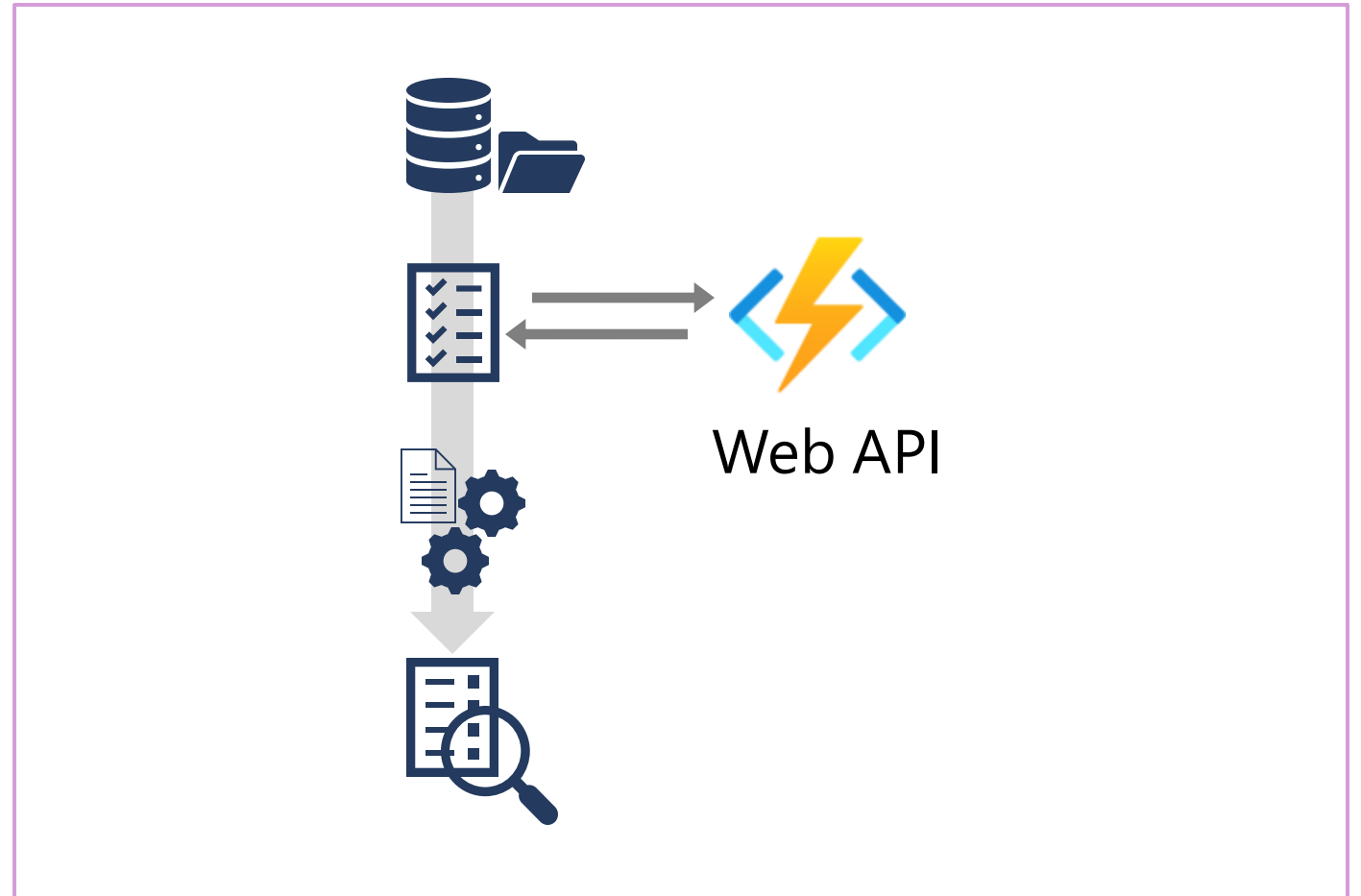
When built-in skills don't provide what you need...

Create a custom skill, for example:

- Integrate Document Intelligence
- Consume an Azure Machine Learning model
- Any other custom logic

Custom skills are implemented as Web APIs

- Commonly Azure Functions



Custom Skill Interfaces

Input Schema

```
{
  "values": [
    {
      "recordId": "<unique_identifier>",
      "data": {
        "<input1_name>": "<input1_value>",
        "<input2_name>": "<input2_value>",
        ...
      }
    },
    {
      "recordId": "<unique_identifier>",
      "data": {
        "<input1_name>": "<input1_value>",
        "<input2_name>": "<input2_value>",
        ...
      }
    },
    ...
  ]
}
```

Output Schema

```
{
  "values": [
    {
      "recordId": "<unique_identifier_from_input>",
      "data": {
        "<output1_name>": "<output1_value>",
        ...
      },
      "errors": [...],
      "warnings": [...]
    },
    {
      "recordId": "< unique_identifier_from_input>",
      "data": {
        "<output1_name>": "<output1_value>",
        ...
      },
      "errors": [...],
      "warnings": [...]
    },
    ...
  ]
}
```

This is a *property bag* of values – it can be a single value or a complex JSON structure

Adding a Custom Skill to a Skillset

Add a Custom.WebApiSkill to the skillset

Specify the URI to your web API endpoint

- Optionally add parameters and headers

Set the context to specify at which point in the document hierarchy the skill should be called

Assign input values

- Usually from existing document fields

Store output in a new field

- Optionally, specify a target field name (otherwise the output name is used)

```
{
  "skills": [
    ...,
    {
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
      "description": "<custom skill description>",
      "uri": "https://<web_api_endpoint>?<params>",
      "httpHeaders": {
        "<header_name>": "<header_value>"
      },
      "context": "/document/<where_to_apply_skill>",
      "inputs": [
        {
          "name": "<input1_name>",
          "source": "/document/<path_to_input_field>"
        }
      ],
      "outputs": [
        {
          "name": "<output1_name>",
          "targetName": "<optional_field_name>"
        }
      ]
    }
  ]
}
```

Exercise – Create a Custom Skill for Azure AI Search



Use an Azure Function to implement a custom skill
Integrate a custom skill into a skillset

Creating a Knowledge Store



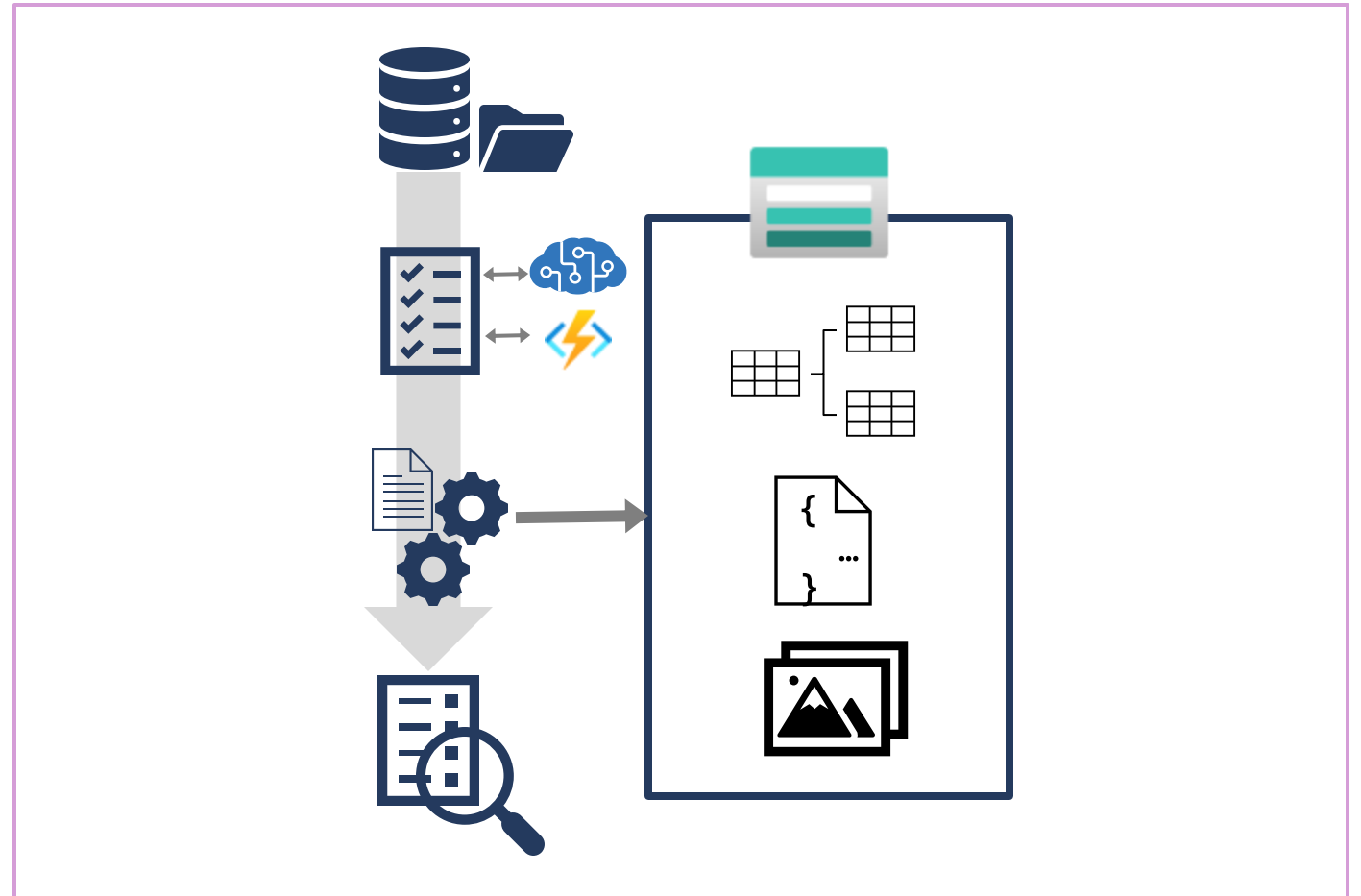
What is a Knowledge Store?

Persisted insights extracted by indexing process

Stored as *projections* in *Azure Storage*

- **Tables:** Relational tables with keys for joining
- **Objects:** JSON structures of document fields
- **Files:** Extracted images saved in JPG format

Used for analysis or integration into data processing workflows



Using the Shaper Skill for Projections

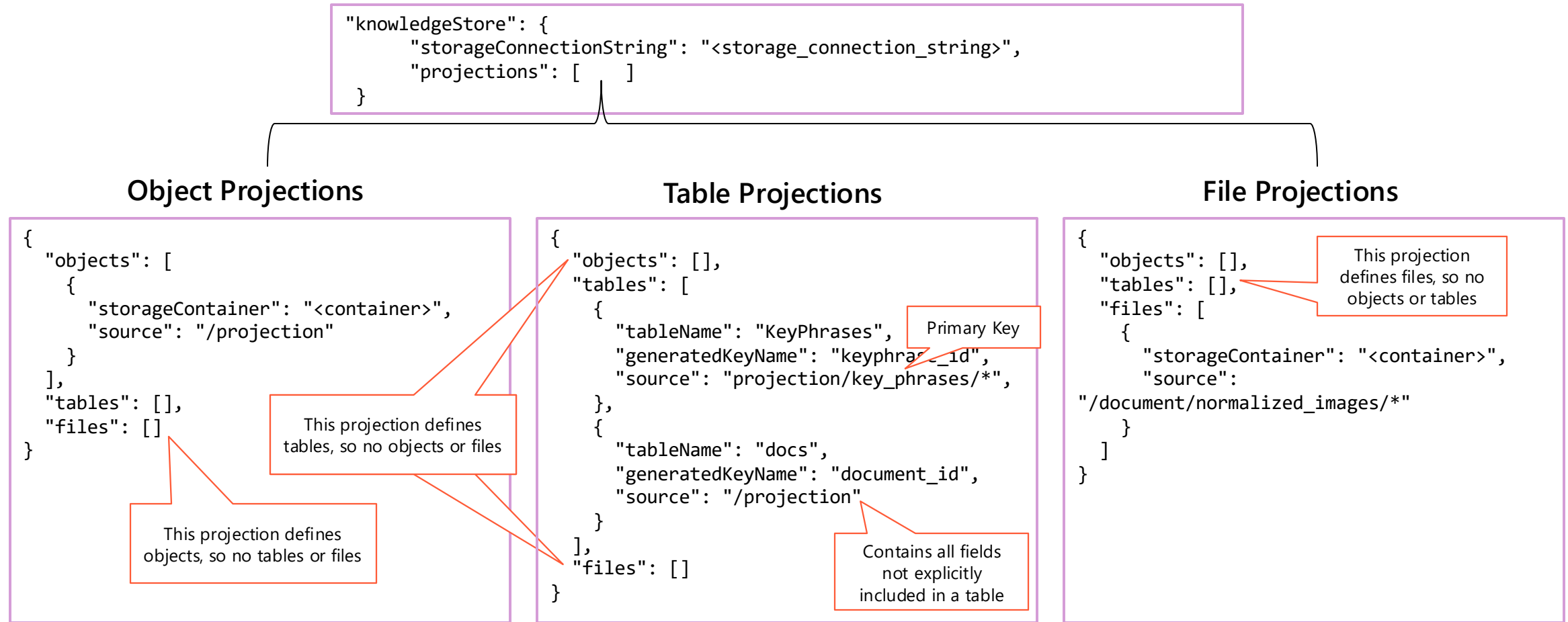
Restructure fields to simplify projections

- Create a JSON object with the fields you want to persist
- Use sourceContext and inputs to map primitives to well-formed JSON objects

```
{
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
  "name": "define-projection",
  "description": "Prepare projection fields",
  "context": "/document",
  "inputs": [
    {
      "name": "url",
      "source": "/document/url"
    },
    {
      "name": "sentiment",
      "source": "/document/sentiment"
    },
    {
      "name": "key_phrases",
      "source": null,
      "sourceContext": "/document/merged_content/keyphrases/*",
      "inputs": [
        {
          "name": "phrase",
          "source": "/document/merged_content/keyphrases/*"
        }
      ]
    }
  ],
  "outputs": [
    {
      "name": "output",
      "targetName": "projection"
    }
  ]
}
```


Implementing a Knowledge Store

Knowledge Store and Projections are defined in the Skillset



Extended interactive exercises – Create a Knowledge Store with Azure AI Search



<https://aka.ms/km-ai-lp>

Knowledge check



1 You want to enrich an index by extracting any geographical locations mentioned in the source data. Which built-in skill should you use?

- ☒ Entity Recognition
- ☐ Key Phrase Extraction
- ☐ Language Detection

2 You have implemented a custom skill as an Azure function. How can you include the custom skill in your indexing process?

- ☐ Add a Merge skill to the skillset to combine output from built-in skills with your custom skill.
- ☒ Add a WebApiSkill to a skillset, referencing the Azure function's URI
- ☐ Add a Shaper skill to the skillset to create a collection of records with unique IDs generated by your custom

3 You want to create a knowledge store that contains JSON representations of the extracted data. What kind of projection should you define?

- ☐ File
- ☐ Object
- ☒ Table



Develop Generative AI Solutions with Azure OpenAI Service



Agenda



- Get started with Azure OpenAI Service
- Develop apps with Azure OpenAI Service
- Apply prompt engineering with Azure OpenAI Service
- Use your own data with Azure OpenAI Service

Get started with Azure OpenAI Service

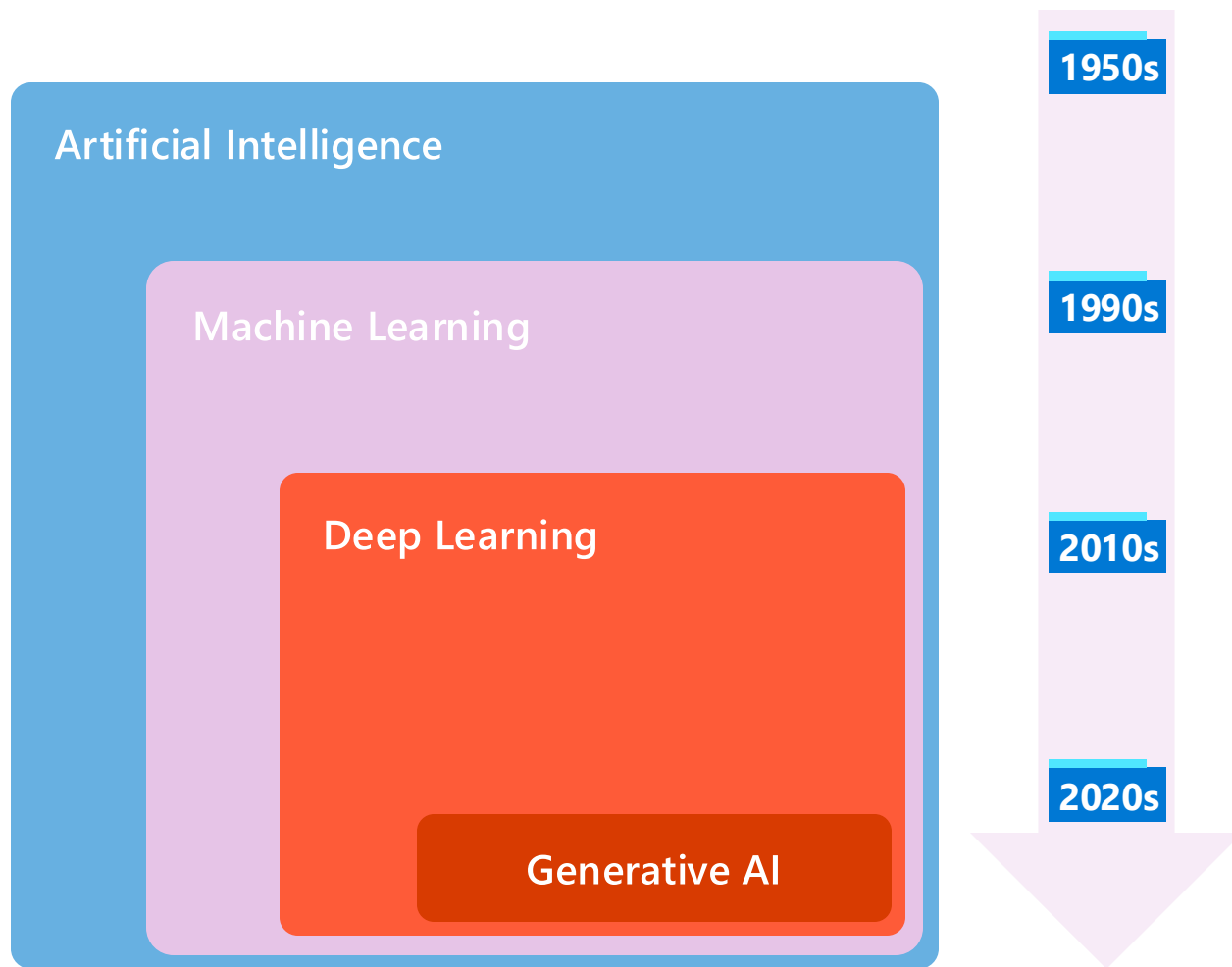


Learning Objectives

After completing this module, you will be able to:

- 1 Describe what generative AI is
- 2 Provision a resource and deploy a model
- 3 Use Azure OpenAI Studio

What is generative AI?



Artificial Intelligence

the field of computer science that seeks to create intelligent machines that can replicate or exceed human intelligence

Machine Learning

subset of AI that enables machines to learn from existing data and improve upon that data to make decisions or predictions

Deep Learning

a machine learning technique in which layers of neural networks are used to process data and make decisions

Generative AI

Create new written, visual, and auditory content given prompts or existing data.

Provision an Azure OpenAI resource in Azure

Deploy a model in Azure OpenAI Studio to use it

1. Apply for access to the Azure OpenAI service:
<https://aka.ms/oaiapply>
2. Create an **Azure OpenAI** resource in the Azure portal

Alternatively, use the Azure CLI

```
az cognitiveservices account create \  
-n MyOpenAIResource \  
-g MyResourceGroup \  
-l eastus \  
--kind OpenAI \  
--sku s0 \  
--subscription subscriptionID
```

[Home](#) > [Azure AI services](#) | [Azure OpenAI](#) >

Create Azure OpenAI ...

1 Basics 2 Network 3 Tags 4 Review + submit

Enable new business solutions with OpenAI's language generation capabilities powered by GPT-3 models. These models have been pretrained with trillions of words and can easily adapt to your scenario with a few short examples provided at inference. Apply them to numerous scenarios, from summarization to content and code generation.

[Learn more](#)

Project Details

Subscription * ⓘ

Resource group * ⓘ

[Create new](#)

Instance Details

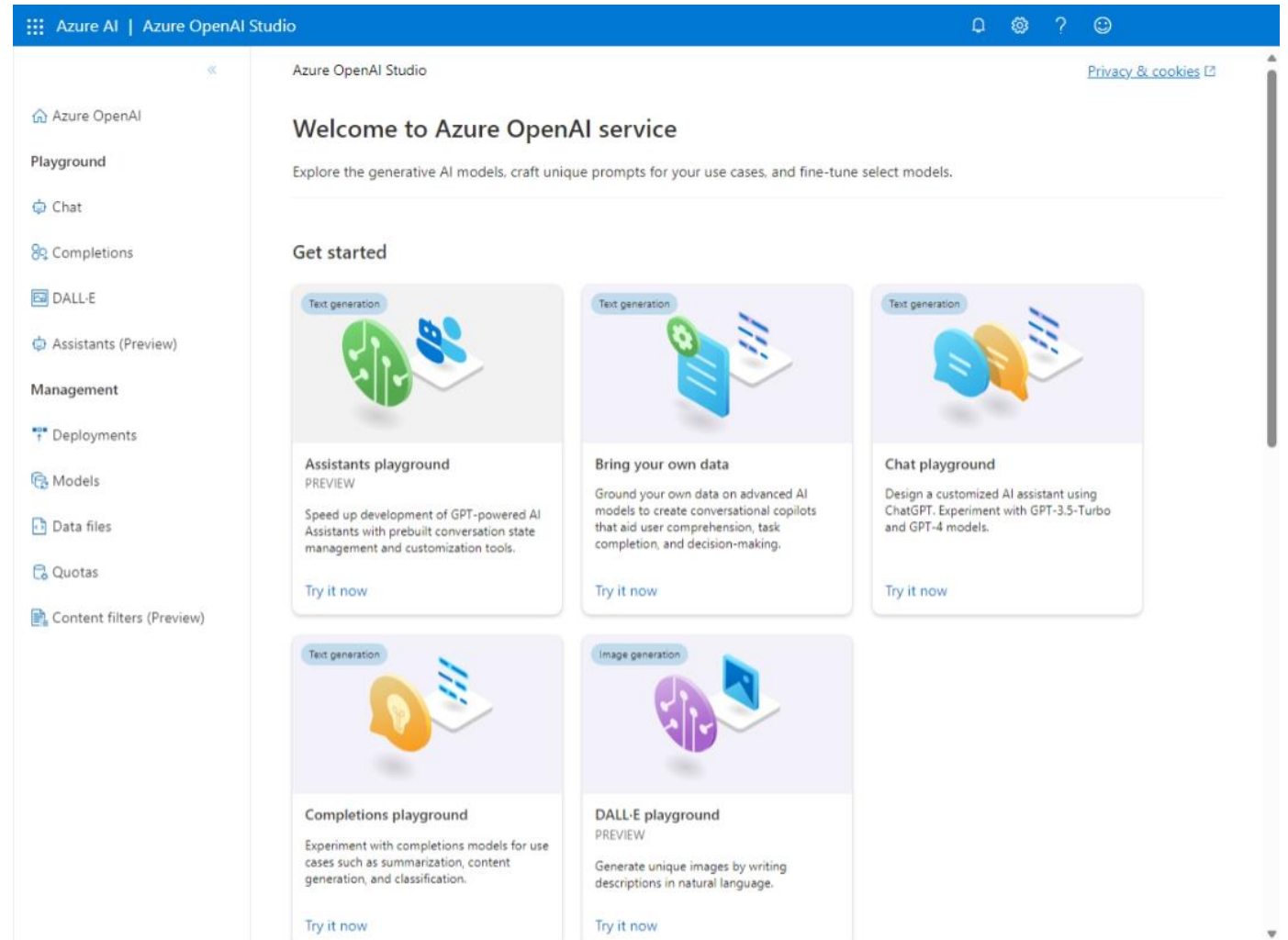
Region ⓘ

Name * ⓘ

Pricing tier * ⓘ

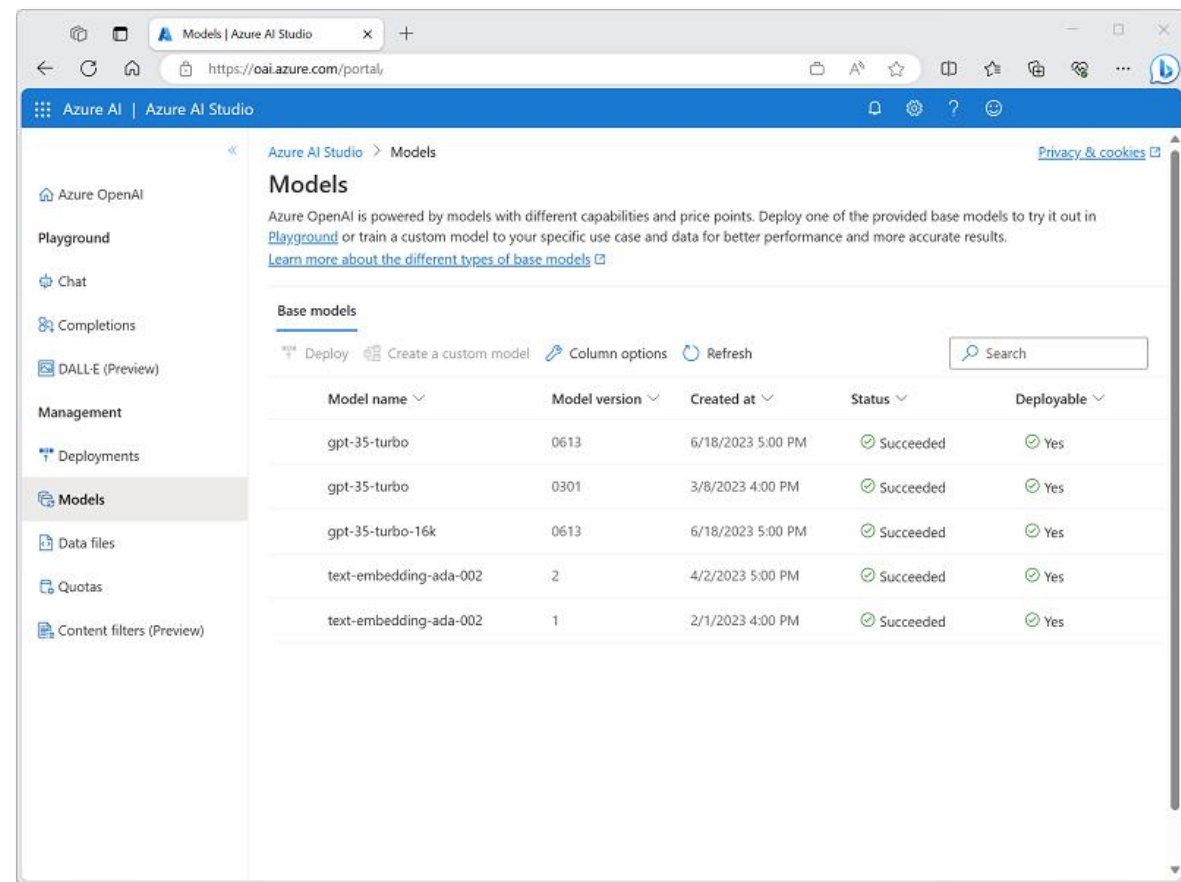
Azure OpenAI Studio

- Web portal for working with Azure OpenAI models:
<https://oai.azure.com/>
- View and deploy base models
- Connect your own data source
- Manage fine tuning and data files for custom models
- Test models in visual playgrounds:
 - **Chat** (GPT-3.5-Turbo and later models)
 - **Completions** (GPT-3 and earlier models)
 - **DALL-E** (Image generations)
 - **Assistants** (Custom and Copilot-like experiences)



Types of generative AI model

Model Family	Description
GPT-4	Newest, most capable chat-based models for language and code generation
GPT-3	Natural language and code-generation models
Embeddings	Models that use embeddings for specific tasks (similarity, text search, and code search)
DALL-E	Image-generation model (<i>preview, restricted regions</i>)

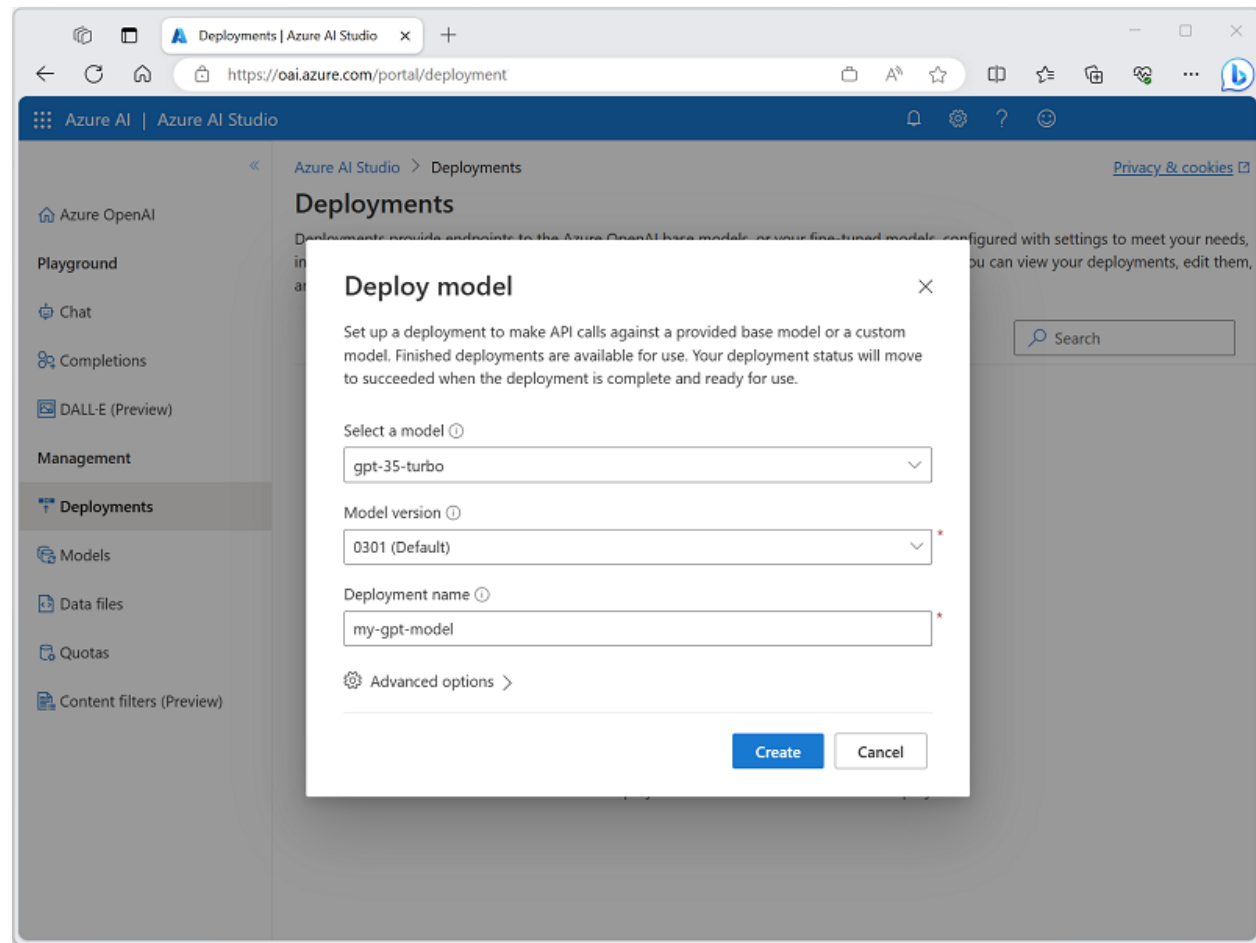


Deploying generative AI models

Deploy a model in Azure OpenAI Studio to use it

- You can deploy one or more instances of each available model
- The number of deployments depends on your quota, which you can see in the portal
- Alternatively, use the Azure CLI

```
az cognitiveservices account deployment create \  
-g myResourceGroupName \  
-n MyOpenAIResource \  
--deployment-name my-gpt-model \  
--model-name gpt-35-turbo \  
--model-version "0301" \  
--model-format OpenAI \  
--scale-settings-scale-type "Standard"
```



Using prompts to get completions from models

Task	Prompt	Completion
Classifying content	Tweet: I enjoyed the training course. Sentiment:	Positive
Generating new content	Write a poem about databases	Databases, oh databases, You keep our information safe, From the small to the large, You store our data in a place.
Transformation/Translation	English: Hello French:	Bonjour
Summarization	Scotland is [<i>long description of Scotland...</i>]	Scotland is [<i>summarized description...</i>]
	Summarize the previous text	
Continuation	One way to grow tomatoes is to	start with seeds...
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	<i>Setup, followed by messages...</i>	<i>A sequence of relevant responses</i>

Testing models in Azure OpenAI Studio playground

The screenshot displays the Azure OpenAI Studio Chat playground interface. The top navigation bar includes the Azure AI logo and the text 'Azure OpenAI Studio'. The left sidebar contains a 'Playground' section with a 'Chat' tab selected, and a 'Management' section with links to 'Deployments', 'Models', 'Data files', 'Quotas', and 'Content filters (Preview)'. The main area is titled 'Chat playground' and features a 'Setup' panel on the left, a central chat area, and a 'Configuration' panel on the right. The 'Setup' panel includes a 'Prompt' section with 'Add your data' and 'Apply changes' buttons, a 'Use a system message template' section with a 'Select a template' dropdown, a 'System message' text area containing 'You are an AI assistant that helps people find information.', and an 'Examples' section with a 'Using examples' box. The central chat area has a 'Start chatting' button and a text input field with the placeholder 'Type user query here. (Shift + Enter for new line)'. The 'Configuration' panel shows 'Deployment' and 'Parameters' tabs, with sliders for 'Max response' (800), 'Temperature' (0.7), 'Top P' (0.95), 'Stop sequence' (Stop sequences), 'Frequency penalty' (0), and 'Presence penalty' (0). It also displays a 'Current token count' of 11/4000 and an 'Input tokens progress indicator'.

Azure AI | Azure OpenAI Studio

Azure OpenAI Studio > Chat playground

Chat playground

Deploy to

Import setup Export setup Show panels

Setup

Prompt Add your data

Apply changes

Use a system message template

Using templates

Use a template to get started, or just start writing your own system message below. Want some tips? [Learn more](#)

Select a template

System message

You are an AI assistant that helps people find information.

Examples

Using examples

Add examples to show the chat what responses you want. It will try to mimic any responses you add here so make sure they match the rules you laid out in the system message.

Start chatting

Test your assistant by sending queries below. Then adjust your assistant setup to improve the assistant's responses.

Type user query here. (Shift + Enter for new line)

Configuration

Deployment Parameters

Max response 800

Temperature 0.7

Top P 0.95

Stop sequence Stop sequences

Frequency penalty 0

Presence penalty 0

Learn more

Current token count 11/4000

Input tokens progress indicator

Demo: Get started with Azure OpenAI Service



Deploy Azure OpenAI resource and model
Test deployed model in playground

Develop apps with Azure OpenAI Service



Learning Objectives

After completing this module, you will be able to:

- 1 Integrate Azure OpenAI into your app
- 2 Use the REST API
- 3 Use language specific SDKs

Integrating Azure OpenAI into your app

Applications submit prompts to deployed models. Responses are completions.

Three REST API endpoints:

- **Completion** - model takes an input prompt, and generates one or more predicted completions
- **Embeddings** - model takes input and returns a vector representation of that input
- **ChatCompletion** - model takes input in the form of a chat conversation (where roles are specified with the message they send), and the next chat completion is generated

ChatCompletion will be the endpoint we focus on for this course

Use **Completion** and **Embeddings** with GPT-3 based models

Use **ChatCompletion** with GPT-3.5-Turbo and later models

Using the Azure OpenAI REST API

Completion Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/completions>

```
{  
  "prompt": "Your favorite Shakespeare  
             play is",  
  "max_tokens": 5  
}
```



```
{  
  "id": "1234...",  
  "object": "text_completion",  
  "created": 1679001781,  
  "model": "gpt-35-turbo",  
  "choices": [  
    {  
      "text": "Macbeth",  
      "index": 0,  
      "logprobs": null,  
      "finish_reason": "stop"  
    }  
  ]  
}
```

Using the Azure OpenAI REST API

Embedding Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/embeddings>

```
{  
  "input": "The food was delicious and  
           the waiter was very  
           friendly..."  
}
```



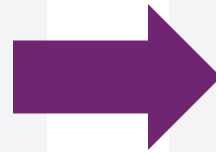
```
{  
  "object": "list",  
  "data": [  
    {  
      "object": "embedding",  
      "embedding": [  
        0.0172990688066482523,  
        ....  
        0.0134544348834753042,  
      ],  
      "index": 0  
    }  
  ],  
  "model": "text-embedding-ada:002"  
}
```

Using the Azure OpenAI REST API

ChatCompletion Endpoint

<https://endpoint.openai.azure.com/openai/deployments/deployment/chat/completions>

```
{
  "messages": [
    { "role": "system",
      "content": "You are an assistant
        that teaches people about AI." },
    { "role": "user",
      "content": "Does Azure OpenAI
        support multiple languages?" },
    { "role": "assistant",
      "content": "Yes, Azure OpenAI
        supports several languages." },
    { "role": "user",
      "content": "Do other Cognitive
        Services support translation?" }
  ]
}
```



```
{
  "id": "unique_id", "object": "chat.completion",
  "created": 1679001781, "model": "gpt-35-turbo",
  "usage": { "prompt_tokens": 95,
             "completion_tokens": 84, "total_tokens": 179 },
  "choices": [
    { "message":
      { "role": "assistant",
        "content": "Yes, other Azure Cognitive
          Services also support translation..." },
      "finish_reason": "stop",
      "index": 0 }
  ]
}
```

Using the Azure OpenAI SDKs

Language specific SDKs are available for use in your applications, in both C# and Python.

Code structure follows a similar pattern for both languages.

Parameters such as *Max Response* and *Temperature* are defined in the chat options.

Both synchronous and asynchronous API versions are available.

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<send request>

<extract response content>
```

Exercise: Integrate Azure OpenAI into your app



Configure your application to connect to Azure OpenAI

Test your app by sending a request to your deployed model

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/mslearn-azure-openai-api>

Apply prompt engineering with Azure OpenAI Service



Learning Objectives

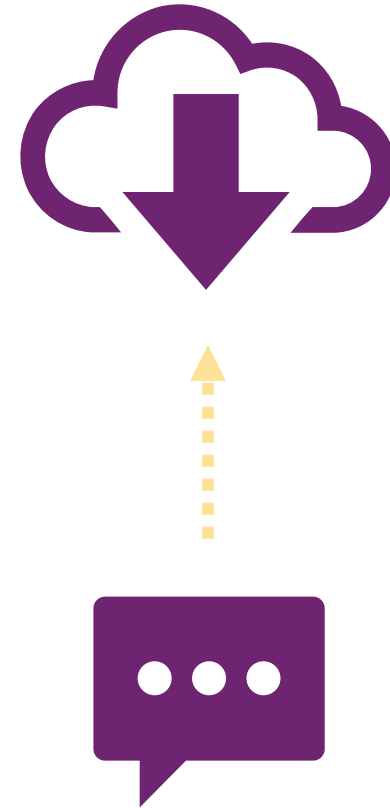
After completing this module, you will be able to:

- 1 Understand what prompt engineering is
- 2 Understand considerations for different endpoints
- 3 Explore different techniques of prompt engineering

What is Prompt Engineering?

Constructing prompts to:

- Maximize relevancy and accuracy of completions
- Specify formatting and style of completions
- Provide conversational context
- Mitigate bias and improve fairness



Providing clear instructions

Write a product description for a new water bottle



Introducing the latest addition to our product line - the innovative and eco-friendly water bottle. Made from high-quality, BPA-free materials, this bottle is designed to keep your beverages at the perfect temperature for hours. Its sleek and stylish design makes it the perfect accessory for any outdoor activity or workout session. With its convenient carrying handle and leak-proof lid, you can take it with you wherever you go, without worrying about spills or leaks. Plus, its easy-to-clean design ensures that you can use it again and again, making it the perfect choice for anyone looking for a reliable and sustainable water bottle.

Write a product description for a new water bottle that is 100% recycled. Be sure to include that it comes in natural colors with no dyes, and each purchase removes 10 pounds of plastic from our oceans



Introducing our newest water bottle, the eco-friendly choice for the conscious consumer. Made from 100% recycled materials and available in natural colors with no dyes, this bottle is not only stylish but also sustainable. Each purchase of this bottle removes 10 pounds of plastic from our oceans, making it a responsible choice for those who care about the environment. Its durable and leak-proof design makes it perfect for outdoor activities, while its easy-to-carry handle makes it convenient to take on the go. Join us in our mission to reduce plastic waste and choose this eco-friendly water bottle for your daily hydration needs.

Primary, supporting, and grounding content

Section markers
(--- or ###)

Primary content to be
summarized, translated, etc.

Supporting content to provide
clarity / specificity

Grounding content to define
scope for questions

Thanks for your interest in our summer school program.

We're looking for your ideas for webinars on topics such as AI, web development, ... *<more...>*

Topics I'm very interested in: AI, webinar dates, submission deadlines

Extract the key points from the above email, and put them in a bulleted list

Artificial Intelligence (AI) has evolved over many years ... *<more...>*

Where and when did the field of AI start?

Cues

- Cues provide a starting point on which the completion builds
- Use cues to set the scope and format for the output
- Cues are particularly useful when generating code. For example:
 - Provide the first keyword in a SQL SELECT statement
 - Define a name and parameters for a Python function

Review: The writing was good but the movie was too long.

Review: The special effects were terrible.

<more>...

Summarize the reviews above.

Most common complaints:

-



- **The movie was too long.**
- **The special effects were terrible.**

Requesting output composition

Write a table in markdown with 6 animals in it, with their genus and species



Animal	Genus	Species
Red Panda	Ailurus	fulgens
African Elephant	Loxodonta	africana
Snow Leopard	Panthera	uncia
Arctic Fox	Vulpes	lagopus
Chimpanzee	Pan	trogloodytes
Siberian Tiger	Panthera	tigris altaica

Using a system message

When using the ChatCompletions API, a system message sets context that can affect completions

```
{"role": "system", "content": "You are a casual, helpful assistant.  
You will talk like an American old western film character."},  
{"role": "user", "content": "Can you direct me to the library?"}
```

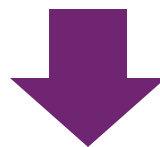


```
{"role": "assistant", "content": "Well howdy there, stranger! The library, huh?  
Y'all just head down the main road till you hit the town  
square. Once you're there, take a left and follow the street  
for a spell. You'll see the library on your right, can't  
miss it. Happy trails!"}
```

Conversation history and *few shot* learning

- Including previous messages helps retain the context and style of a conversation
- Starting a conversation with user-defined messages and responses (*few shot* learning) sets examples for the expected responses

```
{"role": "system", "content": "You are an assistant that evaluates the sentiment of customer feedback."},  
{"role": "user", "content": "That was an awesome experience"},  
{"role": "assistant", "content": "positive"},  
{"role": "user", "content": "I won't do that again"},  
{"role": "assistant", "content": "negative"},  
{"role": "user", "content": "That was not worth my time"},  
{"role": "assistant", "content": "negative"},  
{"role": "user", "content": "You can't miss this"}
```



```
{"role": "assistant", "content": "positive"},
```


Chain of thought

Ask a model to break down its response and explain its reasoning

What sport is easiest to learn but hardest to master? Give a step by step approach of your thoughts, ending in your answer

Step 1: Identify the criteria for "easy to learn" and "hard to master" sports.

For a sport to be considered easy to learn, it should have simple rules and require minimal equipment. For a sport to be considered hard to master, it should require years of practice to perfect and have a large variety of techniques and strategies...

Step 2: Consider different sports that fit these criteria.

Some sports that could be considered easy to learn but hard to master include:

- Tennis: Simple rules, easy to pick up, but requires years of practice to master all the different shots and strategies.
- Golf: Basic swing mechanics are easy to learn but perfecting them takes a lifetime of practice.
- ...

Step 3: Evaluate each sport based on the criteria.

Tennis and golf both require expensive equipment, which could make them less accessible to beginners. In terms of skill ceiling and time required to master, tennis and golf both have a high skill ceiling and require years of practice to perfect....

Step 4: Make a decision based on the evaluation.

Based on the above criteria and evaluation, I would say that tennis is the sport that is easiest to learn but hardest to master. While it may require expensive equipment, it is still accessible to many people. The basic rules and techniques are easy to learn, but mastering all the different shots and strategies takes years of practice...

Exercise: Utilize prompt engineering in your app



Explore prompt engineering techniques

Connect your app to Azure OpenAI and test prompts with increasing prompt engineering

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/openai-prompt-engineering-lab>

Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service



Learning Objectives

After completing this module, you will be able to:

- 1 Understand how RAG using your own data works
- 2 Use the REST API
- 3 Use language specific SDKs

How Azure OpenAI can use your data



Set up your data source

- Use an existing data source, such as an Azure search resource
- Use the Azure OpenAI studio to create that data source, if you don't already have one
- When creating the data source, you can use data already in your account such as blob storage



Configure the studio or your app to connect to that data source

- In the studio, set up the connection by pointing it to the data source
- In your app, specify the data source in the prompt parameters
- Both configurations allow the search resource to augment the prompt

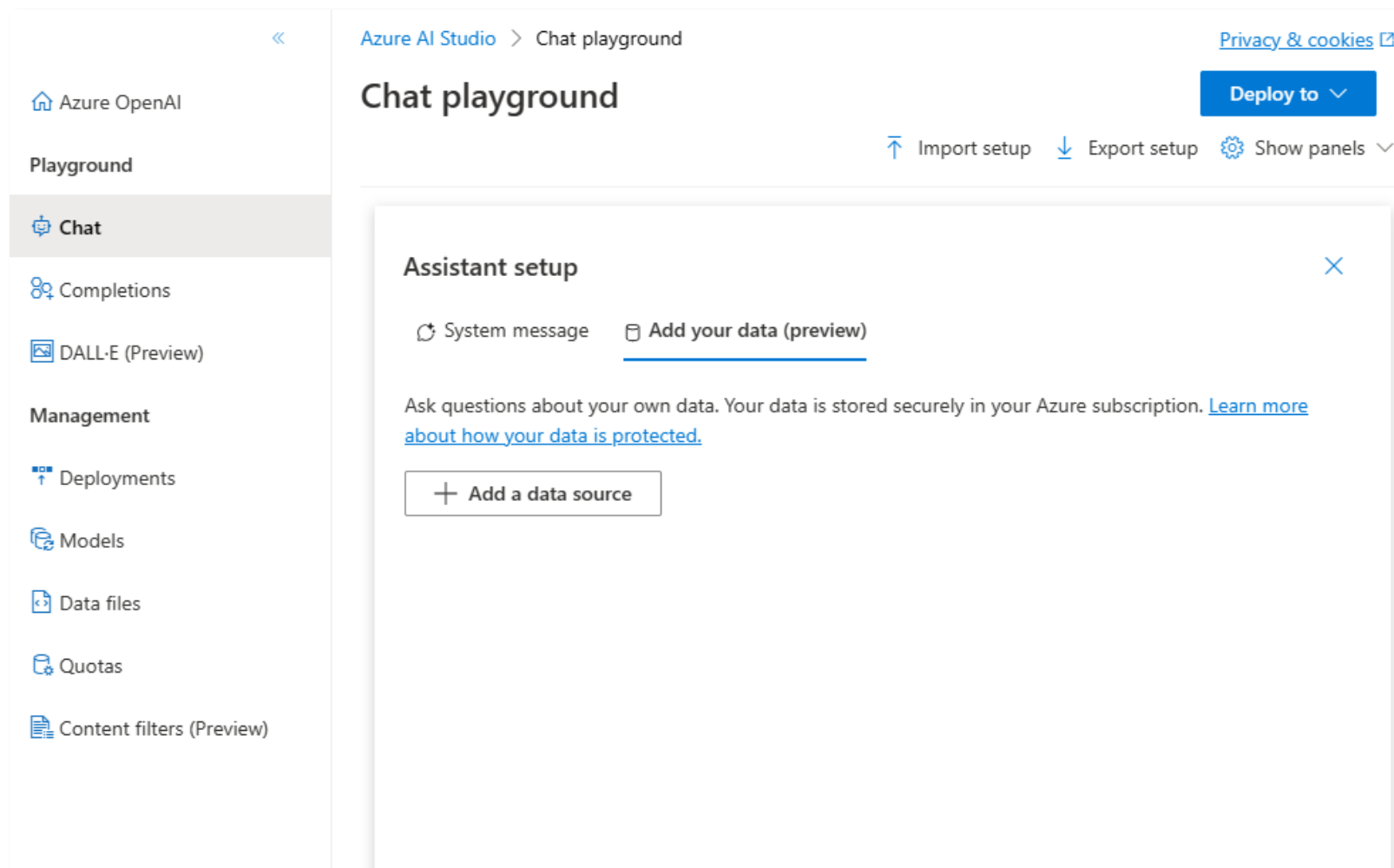


Use the Azure OpenAI model, which now uses your data for grounding

- Chat with the AI models like normal
- If the data source has relevant information about the prompt, it will use that data
- You can specify if the AI model is limited to just your data source

Connect to your data source

- Add your data source in the Chat playground, under Assistant setup
- Use an existing data source, or use that wizard to create a new one
- Once connected, a new chat session will start. Chat like normal, and see how the AI model references that data



Using the Azure OpenAI REST API

Using your own data

<https://endpoint.openai.azure.com/openai/deployments/deployment/chat/completions?api-version=version>

- With each call, you need to specify the data source values, along with the messages array and any other parameters
- Authentication in the data source definition is for your search resource, not your Azure OpenAI resource

```
{
  "data_sources": [
    {
      "type": "azure_search",
      "parameters": {
        "endpoint": "<your_search_endpoint>",
        "index_name": "<your_search_index>",
        "authentication": {
          "type": "system_assigned_managed_identity"
        }
      }
    }
  ],
  "messages": [
    ...
  ]
}
```

Using the Azure OpenAI SDKs

Language specific SDKs are available for use in your applications, in both C# and Python.

Code structure follows a similar pattern for both languages.

Current supported data sources are:

- Azure AI Search
- Azure Cosmos DB for MongoDB vCore
- Plus others in preview, soon to be released GA

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<define data source object to include with request>

<send request>

<extract response content>
```


Exercise: Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service



Set up and connect your data in the chat playground

Configure your app to use your own data for augmenting the prompt

Use the hosted lab environment if provided, or view the lab instructions at the link below:

<https://aka.ms/mslearn-openai-own-data>

Extended interactive exercises



Generate code

Generate images

<https://aka.ms/develop-azure-openai>

Knowledge check



- 1 What is the purpose of providing conversation history to an AI model?**
 - ☐ Providing conversation history to an AI model is irrelevant and has no effect on the AI's performance.
 - ☐ To limit the number of input tokens used by the model
 - ☒ To enable the model to continue responding in a similar way and allow the user to reference previous content in subsequent queries

- 2 Which parameter could you adjust to change the randomness or creativeness of completions?**
 - ☒ Temperature
 - ☐ Frequency penalty
 - ☐ Stop sequence

- 3 You plan to implement a multi-turn conversation with Azure OpenAI. Which endpoint should you use?**
 - ☐ Completions
 - ☒ Chat
 - ☐ Embeddings

AI-102: Designing and Implementing a Microsoft Azure AI Solution

Conclusion



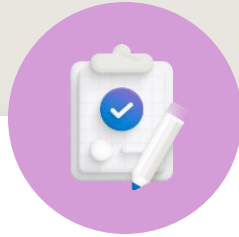
Thank you for attending this course

Here are some reminders



Celebrate your new skills

- Redeem your achievement
- Share with us and your network



Let us know how we did

- Give us your feedback
- Survey will be sent via email



Become Microsoft Certified

- Explore additional resources to help prepare
- Schedule your exam



Tag us social: [#AlwaysLearning](#)

What could be next in your learning journey?

Applied skills with Azure AI

Complete interactive verification of the skills learned in this course and on Microsoft Learn to earn applied skills. Applied skills in Azure AI include:

- Build a natural language processing solution with Azure AI Language
- Build an Azure AI Vision solution
- Create an intelligent document processing solution with Azure AI Document Intelligence
- Develop generative AI with Azure OpenAI Service (released early 2024)

Exam DP-100: Designing and Implementing a Data Science Solution on Azure

Extend your skills in data science and machine learning! Skills covered for this course include:

- Designing and creating a suitable working environment for data science workloads
- Exploring data
- Training machine learning models
- Implementing pipelines
- Running jobs to prepare for production
- Managing, deploying, and monitoring scalable machine learning solutions



Thank you.