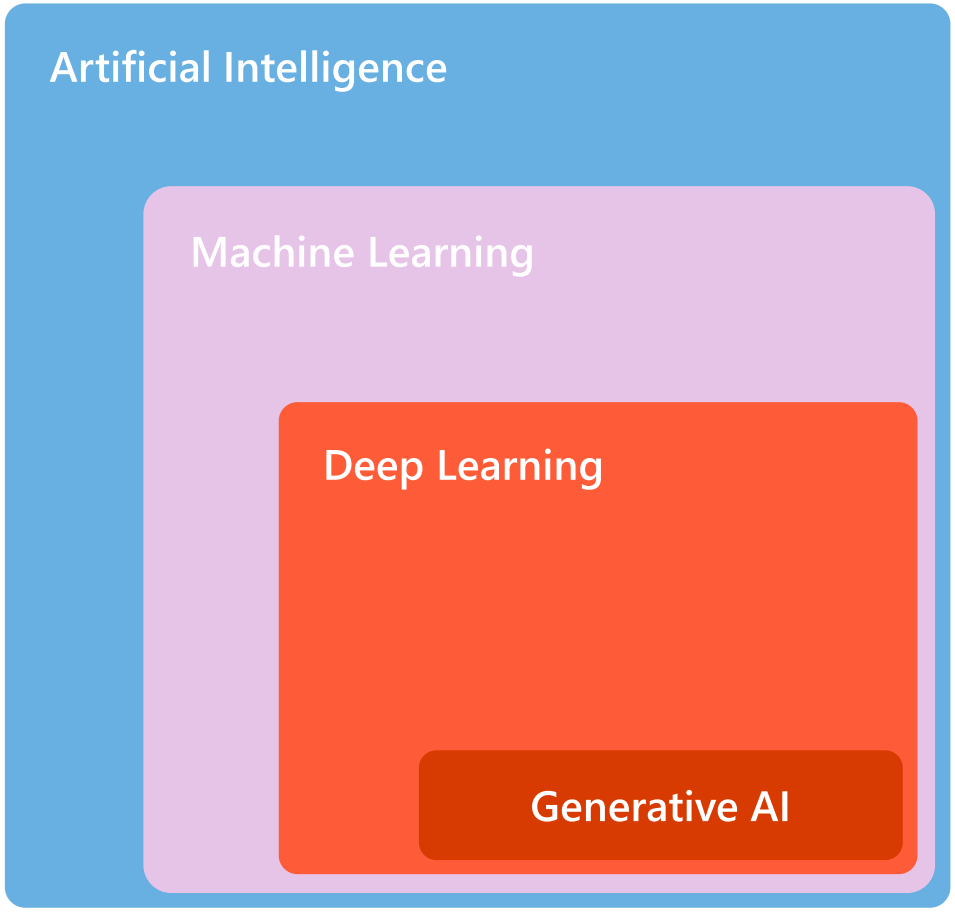
Microsoft

Generative AI & Prompt Engineering with Azure OpenAI

What is generative AI?



1950s

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

1990s

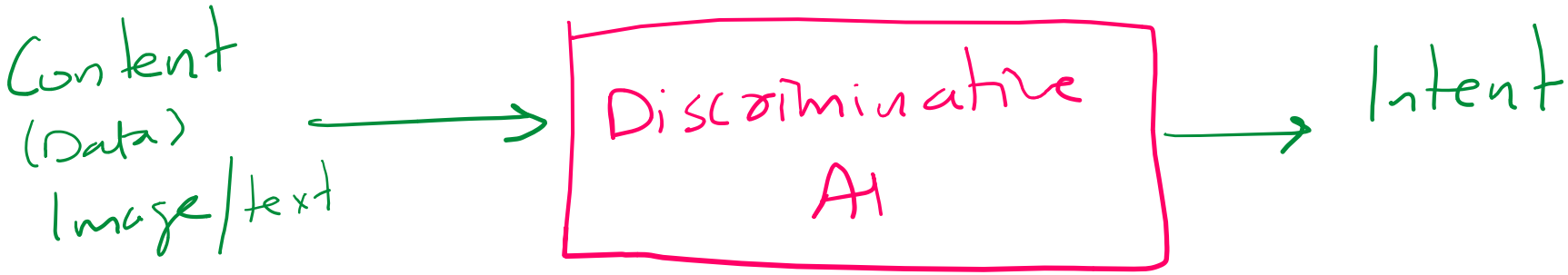
Machine Learning
subset of AI that enables machines to learn from existing data and improve upon that data to make decisions or predictions

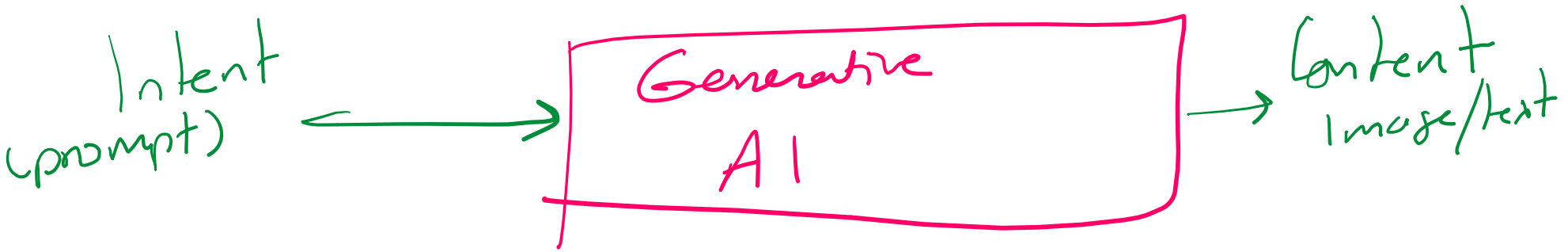
2010s

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

2021

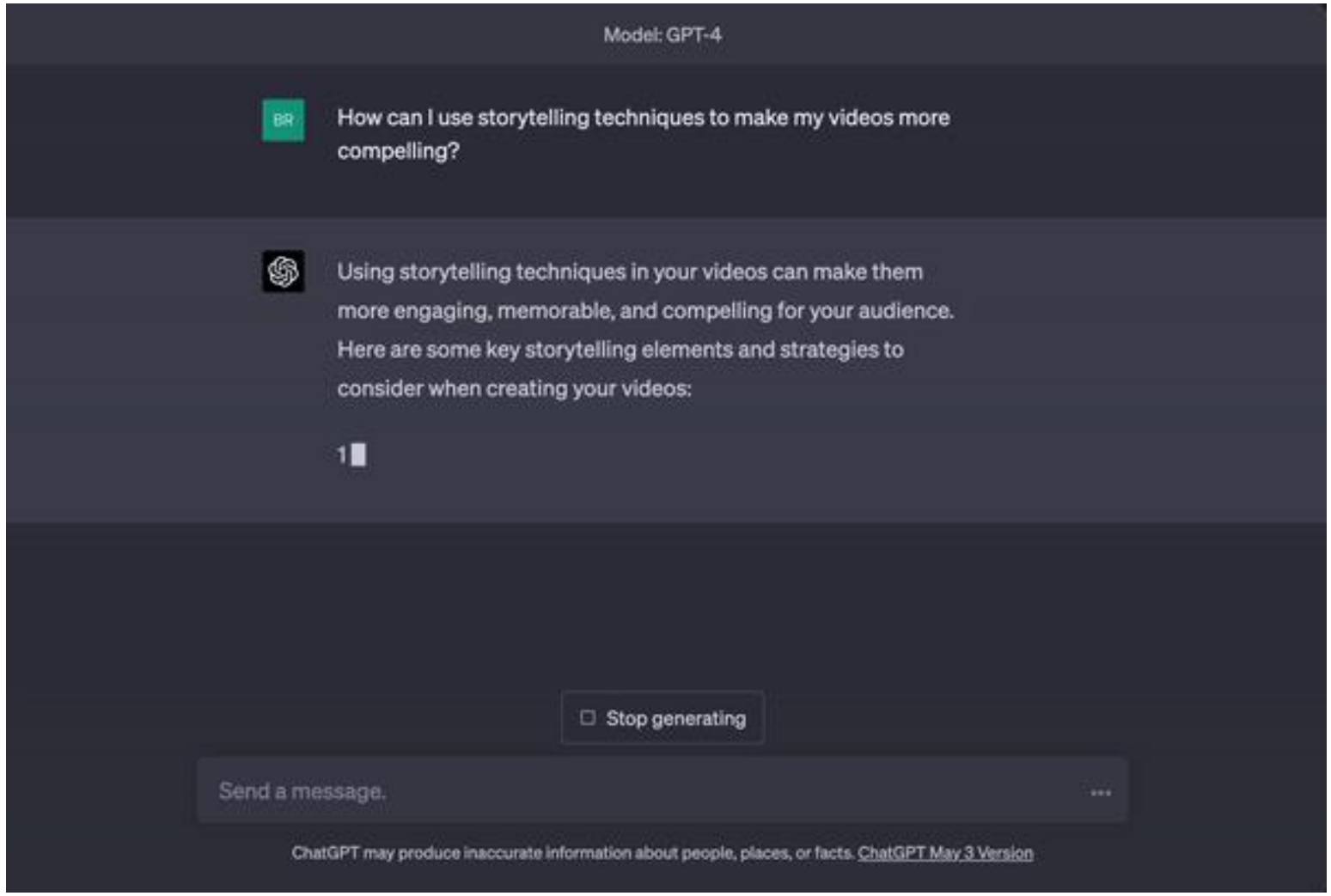
Generative AI
Create new written, visual, and auditory content given prompts or existing data.





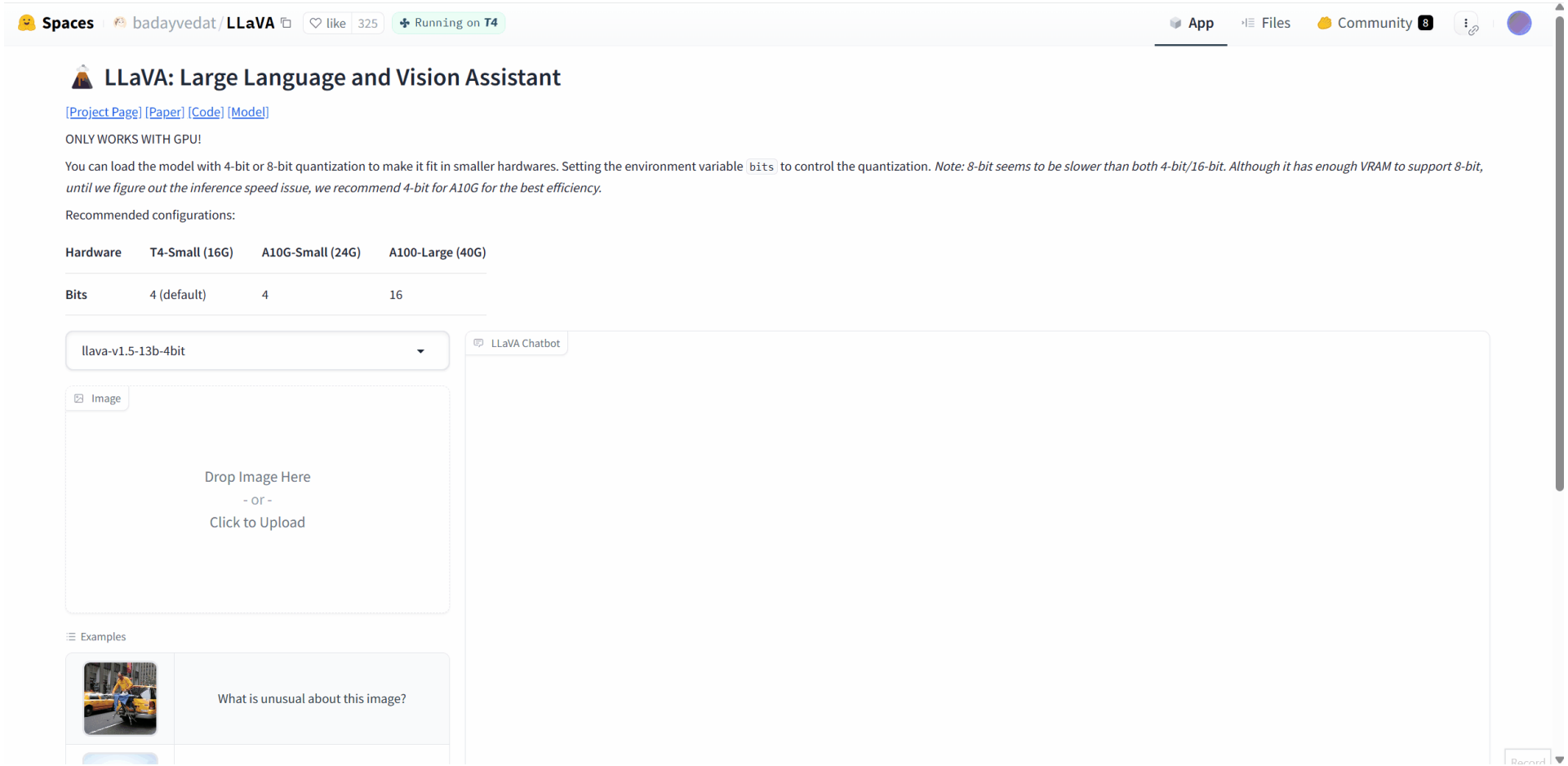
Tasks in Generative AI: Text Generation

- Text Generation: Creating human-like text based on input prompts.
- Latest models/platforms: OpenAI's GPT-4, Google's PaLM & Gemini, Meta's LLaMA.



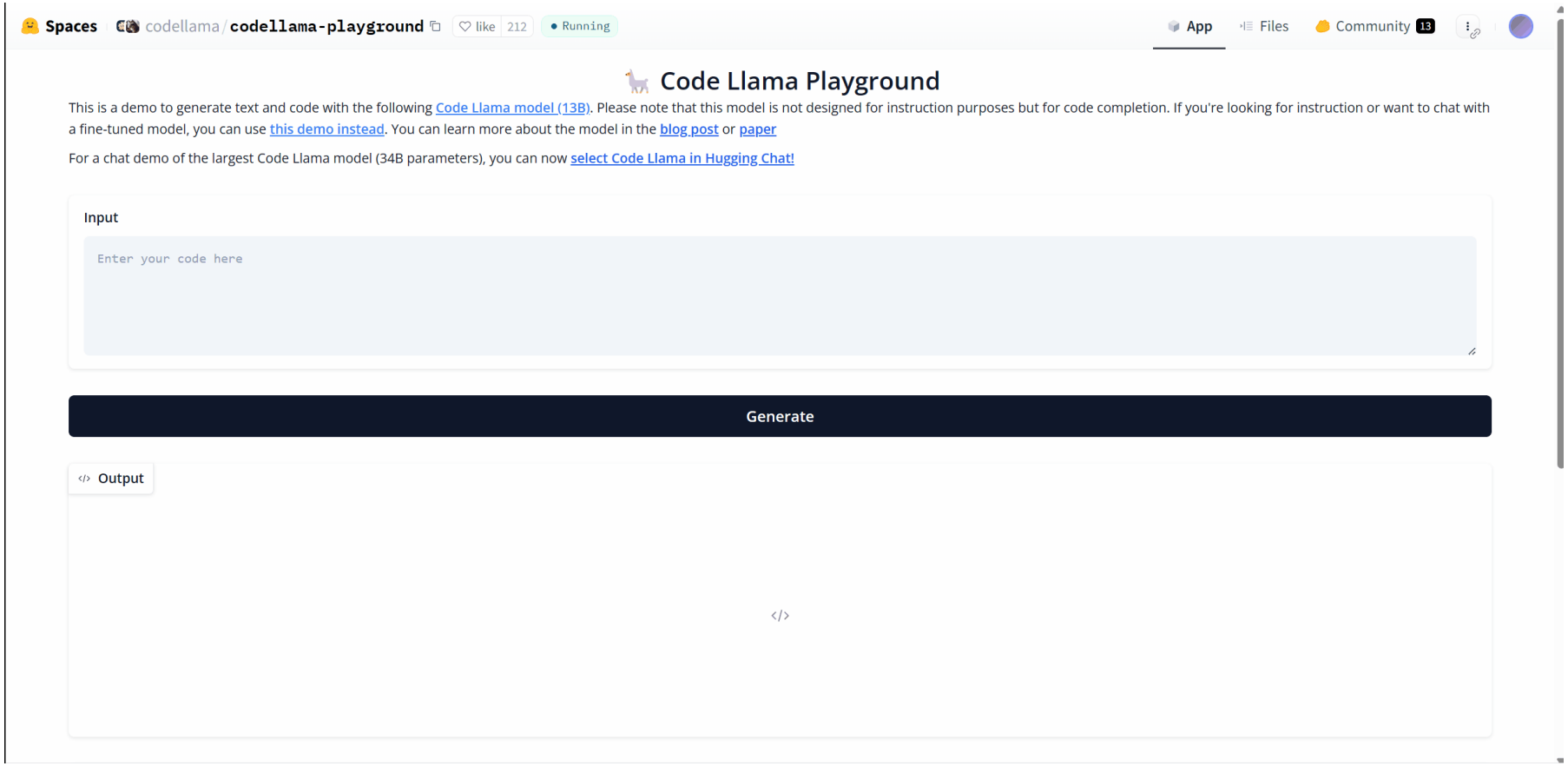
Tasks in Generative AI: Text Generation

Models like GPT-4-vision, llava-v1 can also be used to generate text from images, taking images as prompt.



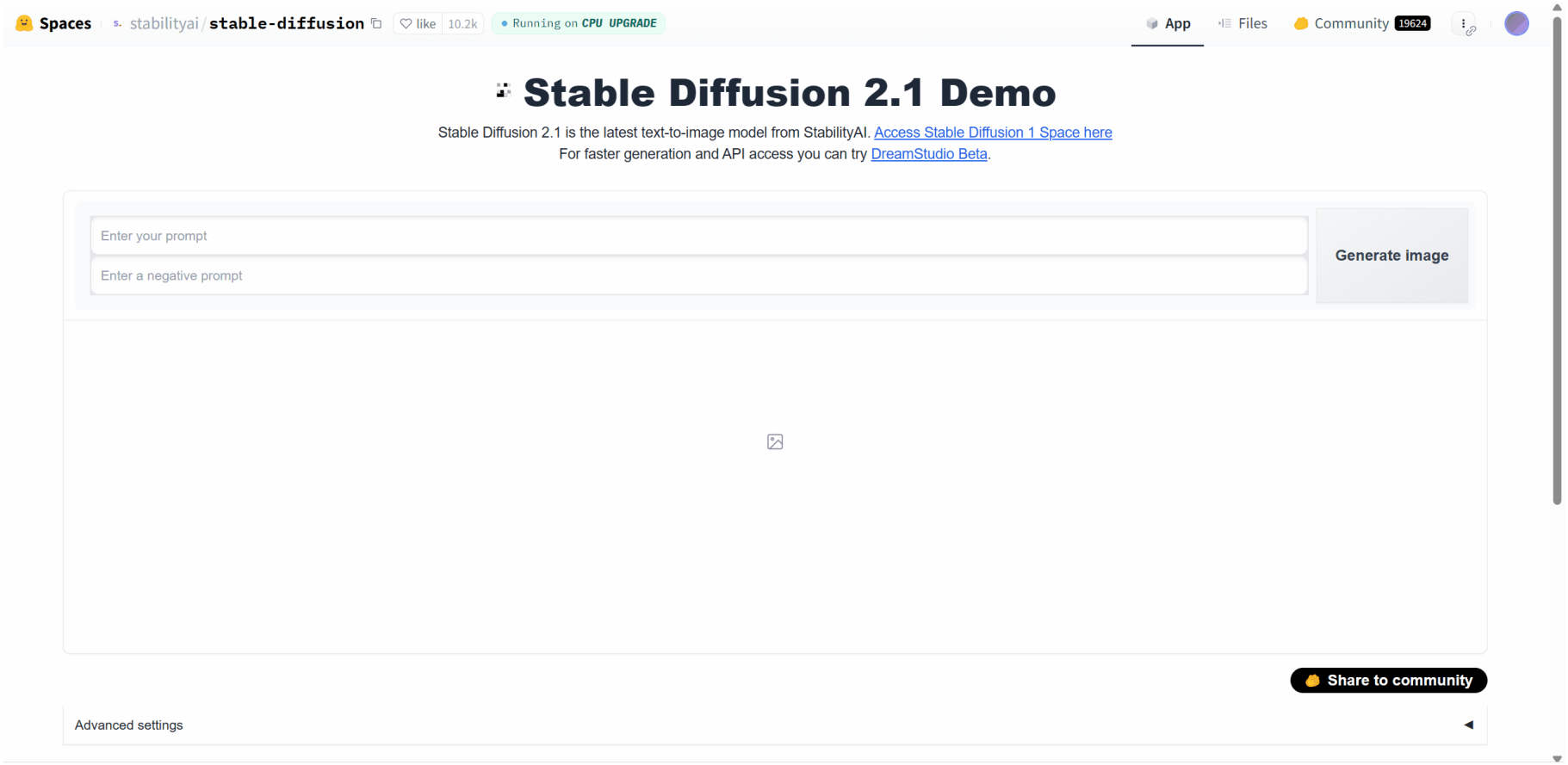
Tasks in Generative AI: Code Generation

- Code Generation: Automatically writing or suggesting code snippets based on a user's instructions or existing code context.
- Latest models/platforms: OpenAI's Codex (which powers GitHub Copilot), Google's AlphaCode, Amazon's CodeWhisperer.



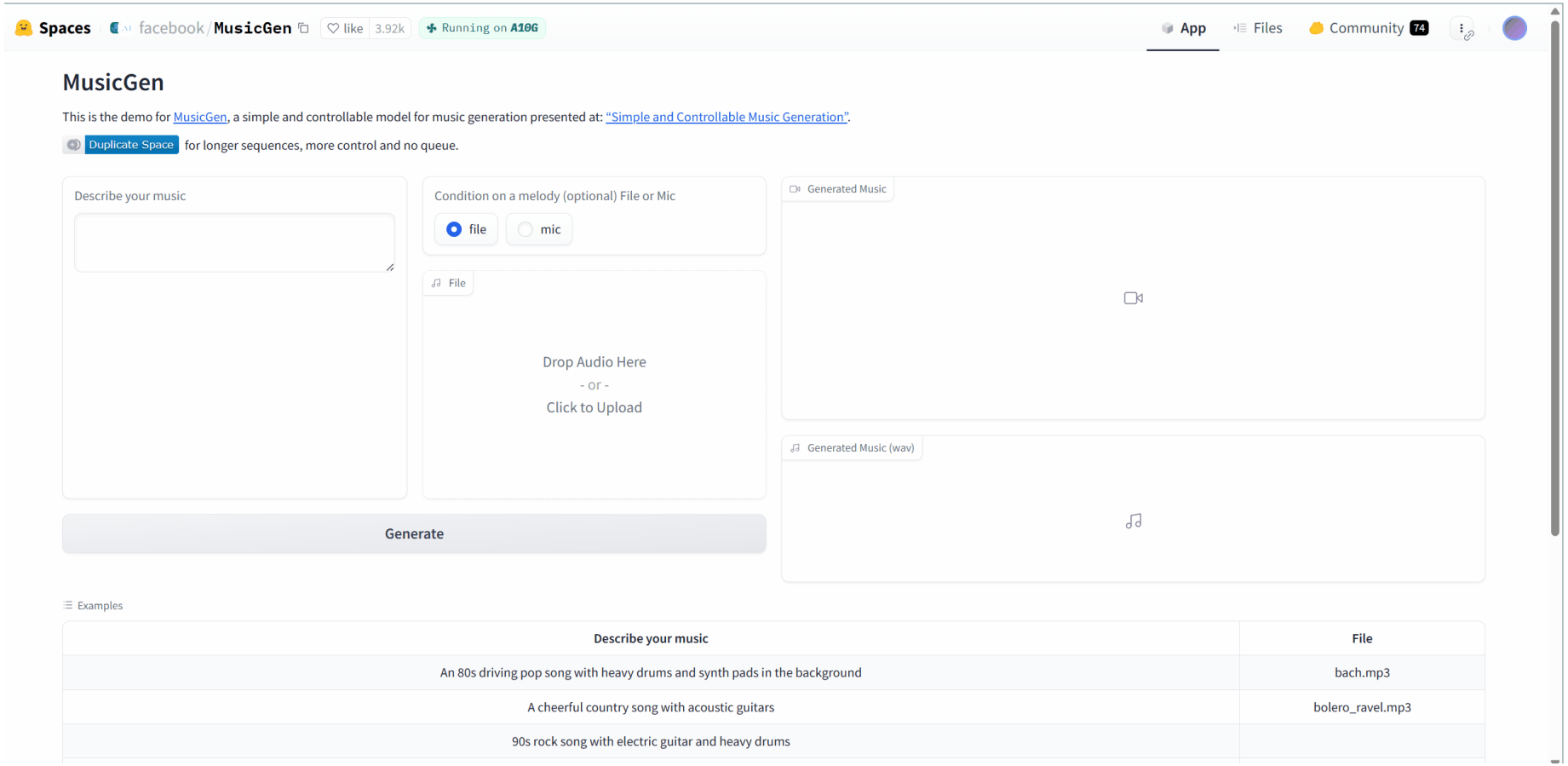
Tasks in Generative AI: Image Generation

- Image Generation: Generating photorealistic or artistic images from textual descriptions.
- Latest models/platforms: OpenAI's DALL-E 2, Stability AI's Stable Diffusion, Google's Imagen.



Tasks in Generative AI: Audio Generation

- Audio Generation: Producing synthetic speech, music, or other sound effects.
- Latest models/platforms: Google's AudioLM, OpenAI's Jukebox, Descript's Overdub.



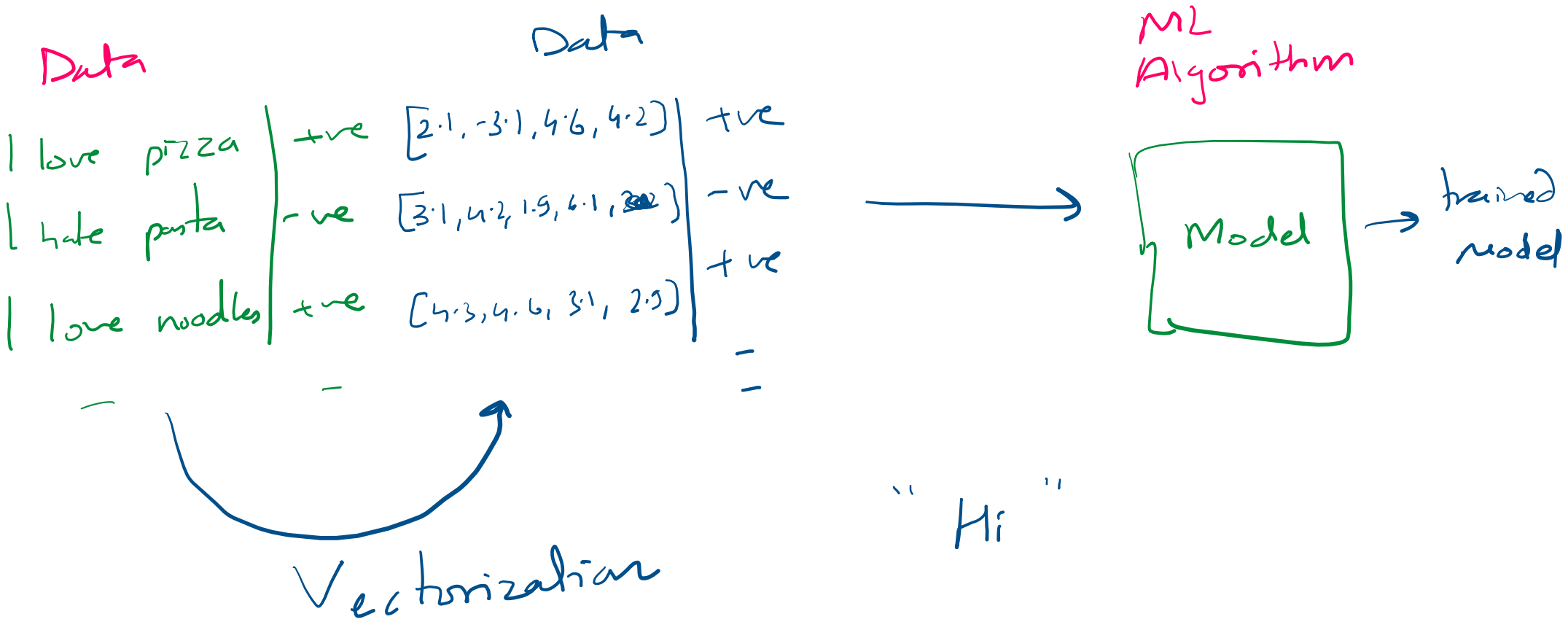
Tasks in Generative AI: Video Generation

- Video Generation: Creating video clips from scratch or modifying existing videos with generative methods.

- Latest models/platforms: OpenAI’s Sora, Meta's Make-A-Video, Google's DreamFusion, Synthesia for AI video production.

Prompt: A young man at his 20s is sitting on a piece of cloud in the sky, reading a book.

Large Language Models



Large Language Models

(Word2vec)

③ → vector size

① I love python coding

② I like python programming.

Unique → [I, like, love, python, Programming, coding]

x	y
I	love
love	Python
love	I
Python	coding
Python	love
I	like
like	I
like	Python
Python	Programming
Python	like

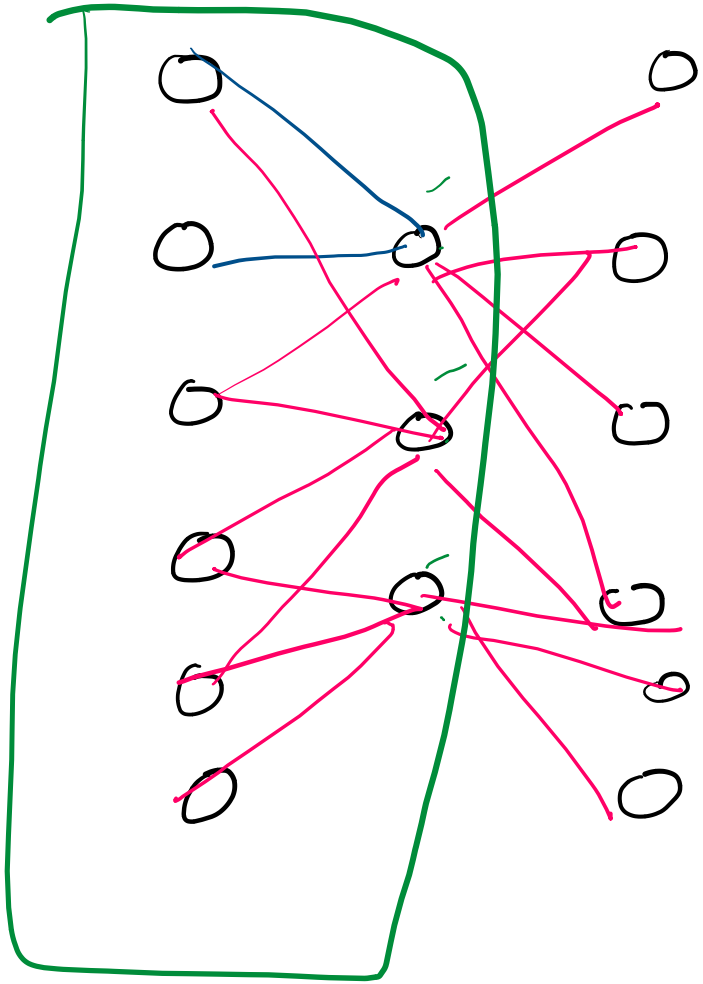
x → 100000

Python → 001000

y

001000

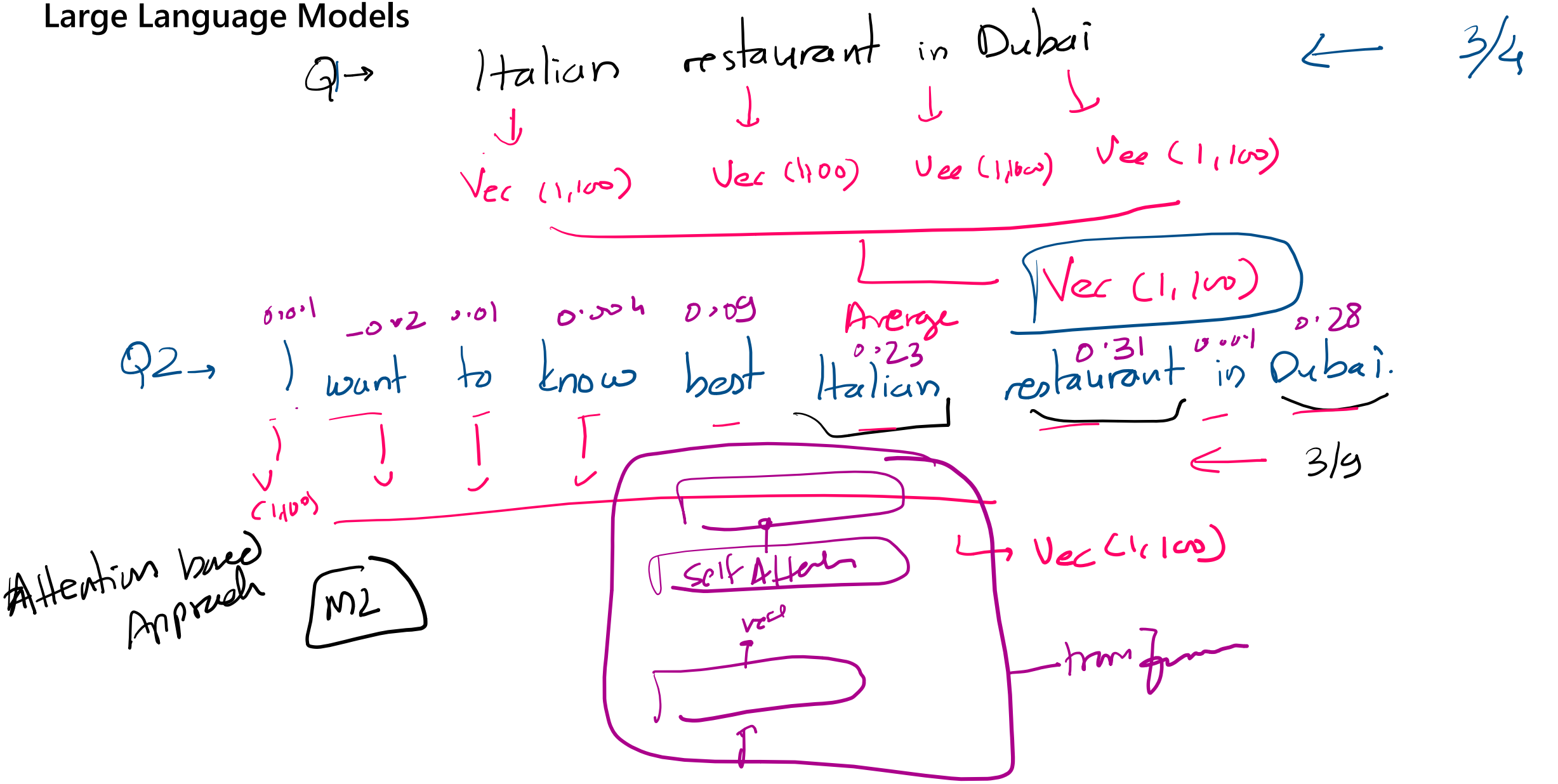
000000



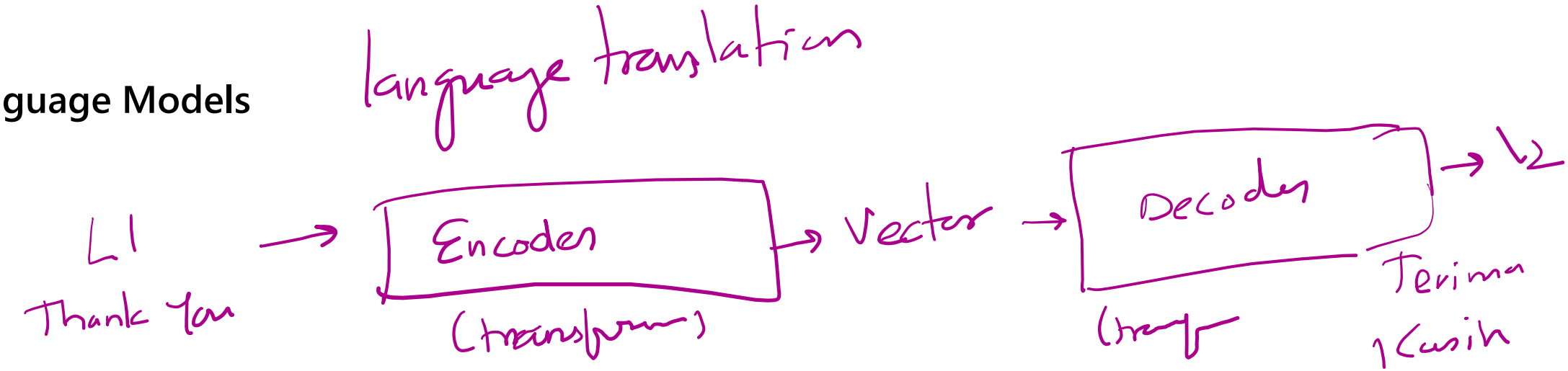
Vector model

2012 - 2016

Large Language Models



Large Language Models



Large Language Models (LLMs)

Type of AI that can process and produce natural language text.

Learns from a massive amount of data gathered from different sources.

A foundation model refers to a specific instance or version of an LLM. Example - GPT-3, GPT-4.

Azure OpenAI Service

Azure OpenAI Service

Deployed in your Azure subscription, secured by you, and tied to your datasets and applications

Large, pretrained AI models to unlock new scenarios

AI models, some custom-tunable with your data and hyperparameters

Built-in responsible AI to detect and mitigate harmful use

Enterprise-grade security with role-based access control (RBAC) and private networks

Top use cases

Business Problem	**Productivity is lagging**	**Need for process Automation**	**Degraded Customer Experience**	**Creating Content is Time Consuming**
Business Needs	**Increase Productivity**	**Automate Processes**	**Improve Customer Experience**	**Build Creative Content**
Solutions	• Conversational Search/Knowledge Insights • Code Generation and Documentation • Trend Forecasting • Report Summarization & Generation	• Document Processing • Workflow Management • Fraud Detection • Supply Chain Optimization	• Intelligent Contact Center • Agent/Employee Assistance • Virtual Assistance • Call Analytics • Call Summarization	• Marketing/Sales Content Generation • Personalized Content Generation • Product Design & Development • Digital Art
What can Generative AI Do?	**Generate New Revenue Streams** **Deliver Differentiated Customer Experiences** **Modernize Internal Processes**			

Top capabilities and scenarios

1.0

Content generation

Summarization

Code generation

Semantic search

2.0

Examples of advanced use cases

Telecommunications

Media Workflows, Cross Content Linking, Content Creation for Media, Speech Analytics, Analytics for B2C Contact Center, Cognitive Contact Center, Skilling Automation, Realtime Speech Transcriptions & Summarization

Manufacturing & Industrials

ChatGPT Enabled Technical Support, Customer Sentiment Analysis, Customer Service Knowledge Mining, Digital Proposal Assistant, Customer Journey Analytics, Consumer Insights Advanced Analytics, Records Summarization, Anomaly Detection, Virtual Agents with Copilot

Automotive, Mobility & Transportation

Marketing Content Generation, Contextual Contact Center, Customer Feedback Loop, Smart Incident Manager, Customer Comms, Text Summarization & Analytics

Customers Bringing it to Life

STRABAG · Coca-Cola · KPMG · Icertis · ODP · CARMAX · zammo.ai

Components and key concepts of Azure OpenAI

Azure Open AI - Components

Pre-trained generative AI models

Customization capabilities

Built-in tools to detect and mitigate harmful use cases

Enterprise-grade security with RBAC

Key concepts

Prompts & Completions

Tokens

Resources

Deployments

Prompt engineering

Models

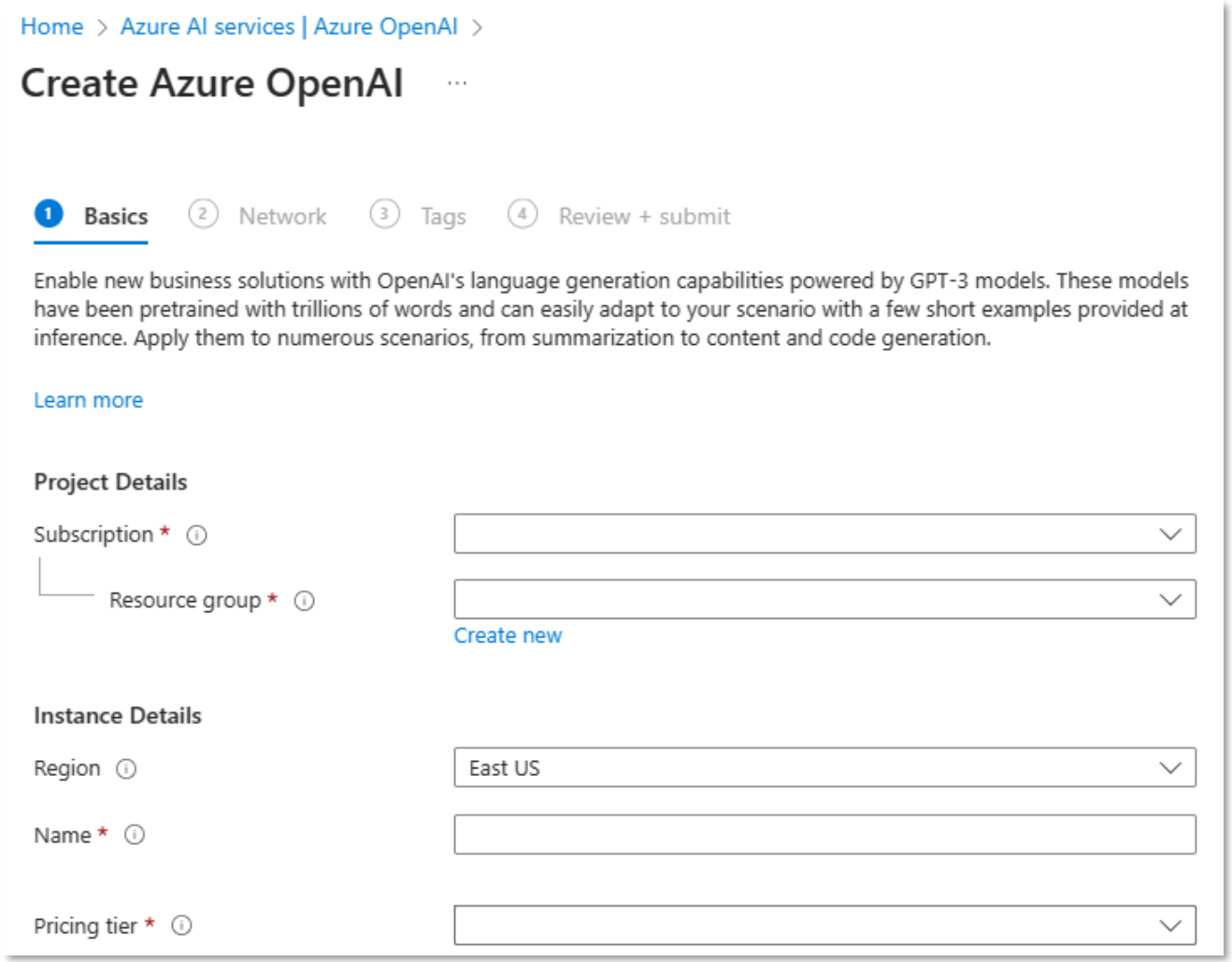
Provision an Azure OpenAI resource in Azure

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

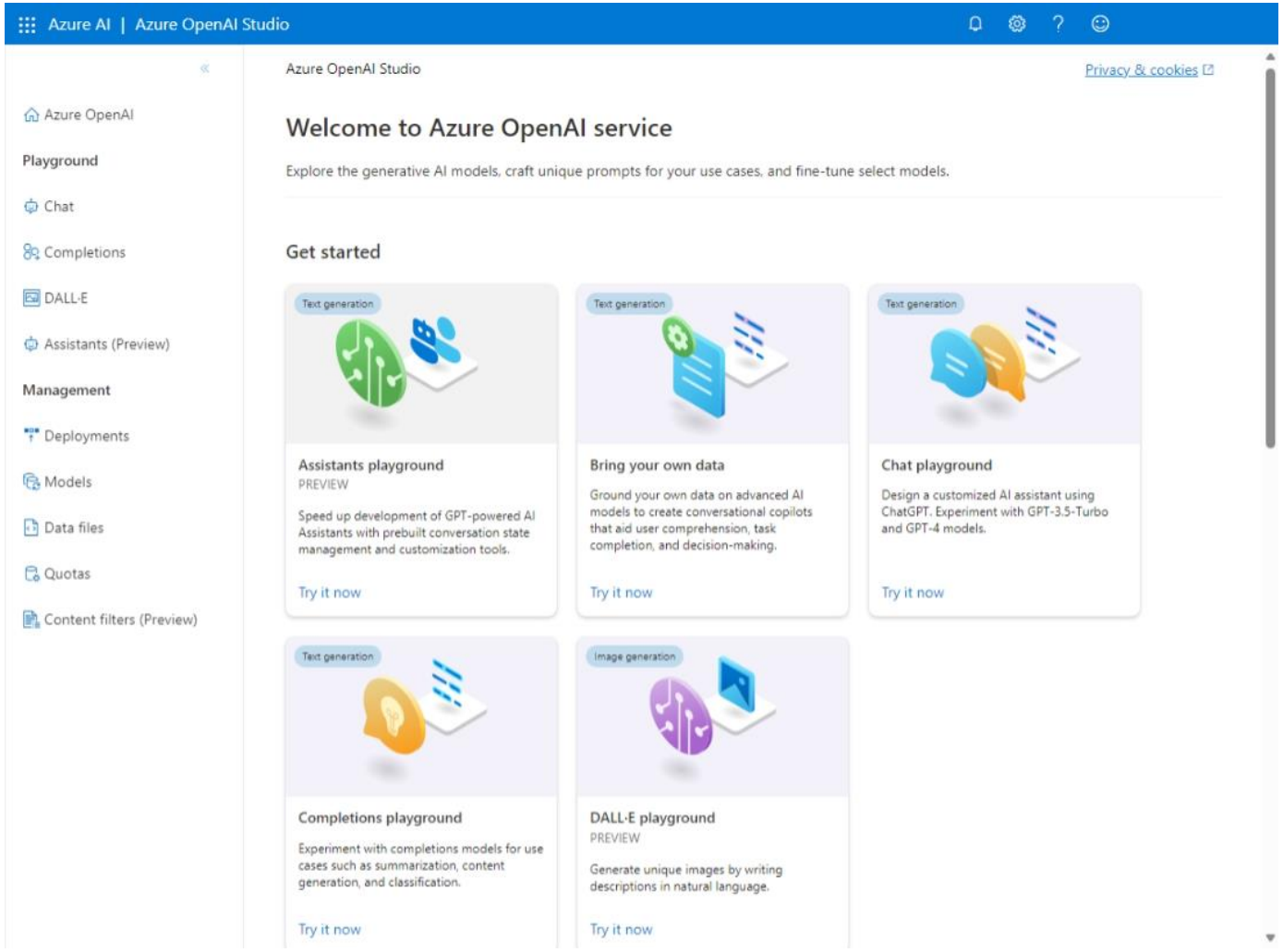
Alternatively, use the Azure CLI

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



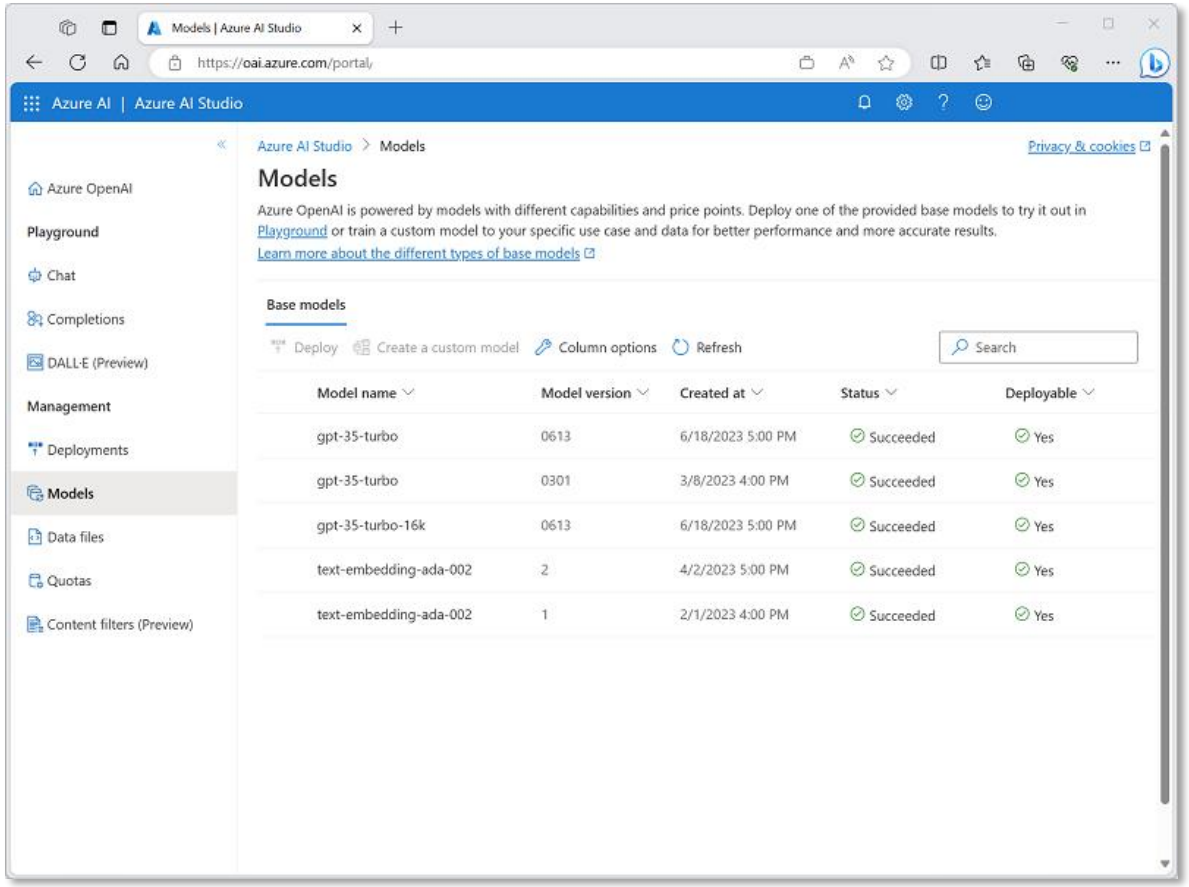
Azure OpenAI Studio

- 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-3.5	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

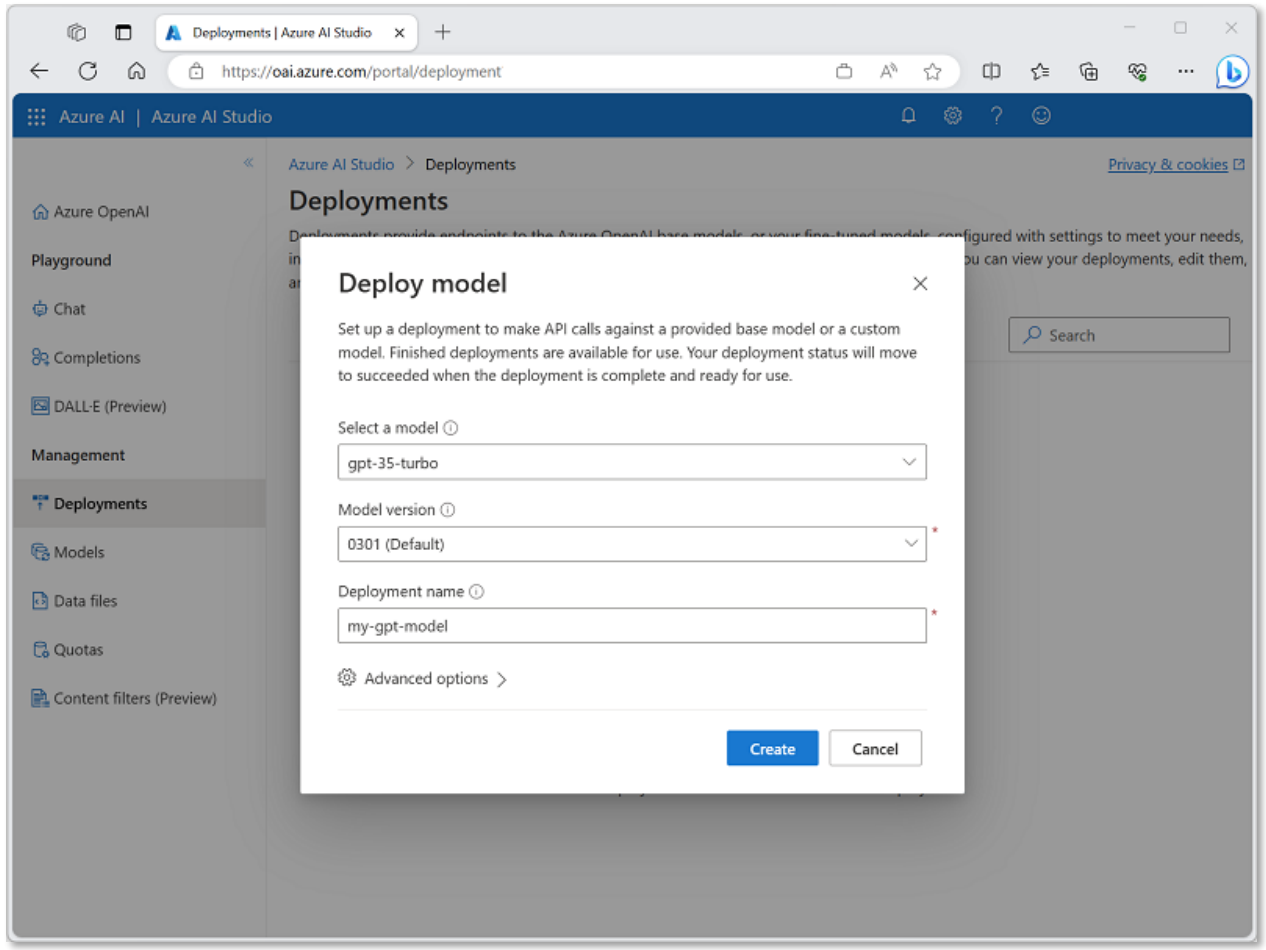


Deploying generative AI models

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

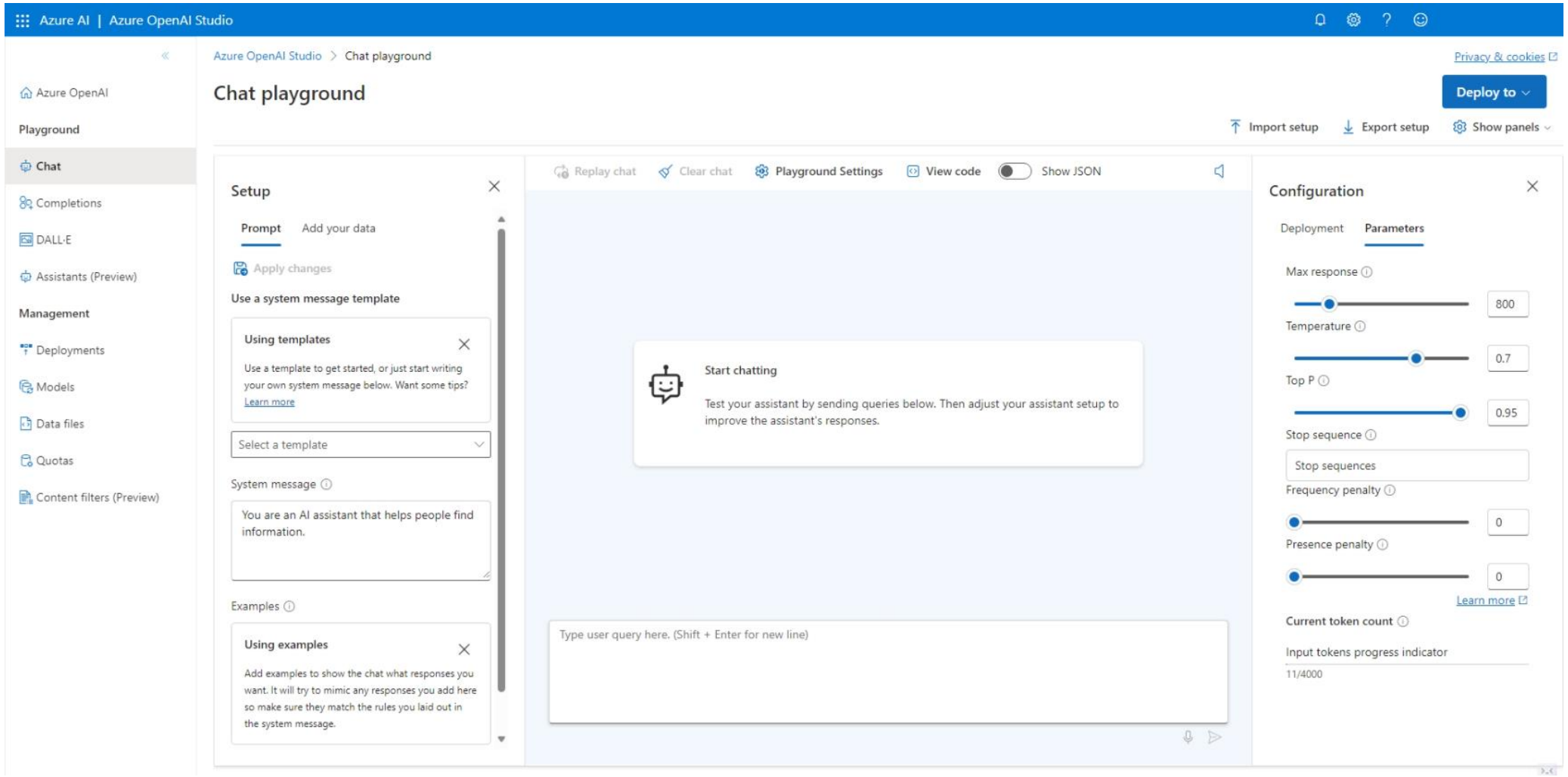
```
az cognitiveservices account deployment create \
    -g myResourceGroupName \
    -n MyOpenAIResource \
    --deployment-name my-gpt-model \
    --model-name gpt-35-turbo \
    --model-version "0301"  \
    --model-format OpenAI \
    --scale-settings-scale-type "Standard"
```



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 [*long description of Scotland…*] Summarize the previous text	Scotland is [*summarized description…*]
Continuation	One way to grow tomatoes is to	start with seeds…
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	*Setup, followed by messages…*	*A sequence of relevant responses*

Testing models in Azure OpenAI Studio playground



Exercise: Get started with Azure OpenAI Service

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

https://aka.ms/mslearn-get-started-azure-openai

Knowledge check

1 **What do you need in order to test a generative AI model using the Azure OpenAI Service Studio?**

- ❑ A deployed model name and Azure command line interface
- ❑ An Azure OpenAI resource and an Azure Cognitive Services resource
- ☑ An Azure OpenAI resource, a deployed model, and a playground

2 **Which parameter could you adjust to change the randomness or creativeness of completions**

- ☑ Temperature
- ❑ Frequency penalty
- ❑ Stop sequence

3 **Which Azure OpenAI Studio playground is able to support conversational interactions that consist of a sequence of messages?**

- ❑ Completions
- ☑ Chat
- ❑ Bot

Use Azure OpenAI APIs in your applications

Integrating Azure OpenAI into your app

Applications submit prompts to deployed models. Responses are completions.

- **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-35-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..."
}
```

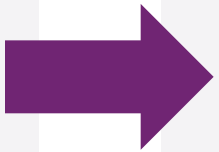
```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "embedding": [
        0.01729906880664825 23,
        ....
        0.01345443488347530 42,
      ],
      "index": 0
    }
  ],
  "model": "text-embedding-ada:002"
}
```

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

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

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

Knowledge check

1 **Which REST endpoint should you use to interact with a GPT-4 model?**

- ❑ Completion
- ❑ Embeddings
- ☑ ChatCompletion

---

2 **When using the .NET SDK, which method should you use to call the ChatCompletion API?**

- ❑ ChatMessage()
- ☑ GetChatCompletions()
- ❑ GetCompletions()

---

3 **When using the Python SDK, which method should you use to call the ChatCompletion API?**

- ❑ openai.ChatCompletion.get()
- ☑ openai.ChatCompletion.create()
- ❑ openai.chat.complete()

Prompt engineering

Model completes the task by predicting the most probable next piece of text.

Three main approaches:

Few-shot – user includes several examples

One-shot – user provides one example

Zero-shot – no examples are provided

```
Convert the questions to a command:
Q: Ask Constance if we need some bread.
A: send-msg `find constance` Do we need some bread?
Q: Send a message to Greg to figure out if things are ready for Wednesday.
A: send-msg `find greg` Is everything ready for Wednesday?
Q: Ask Ilya if we're still having our meeting this evening.
A: send-msg `find ilya` Are we still having a meeting this evening?
Q: Contact the ski store and figure out if I can get my skis fixed before I leave c
A: send-msg `find ski store` Would it be possible to get my skis fixed before I lea
Q: Thank Nicolas for lunch.
A: send-msg `find nicolas` Thank you for lunch!
Q: Tell Constance that I won't be home before 19:30 tonight – unmovable meeting.
A: send-msg `find constance` I won't be home before 19:30 tonight. I have a meeting
Q: Tell John that I need to book an appointment at 10:30.
A:
```

A few-shot prompt where multiple examples are provided. The model will generate the last answer

Models

GPT-4

GPT-3.5

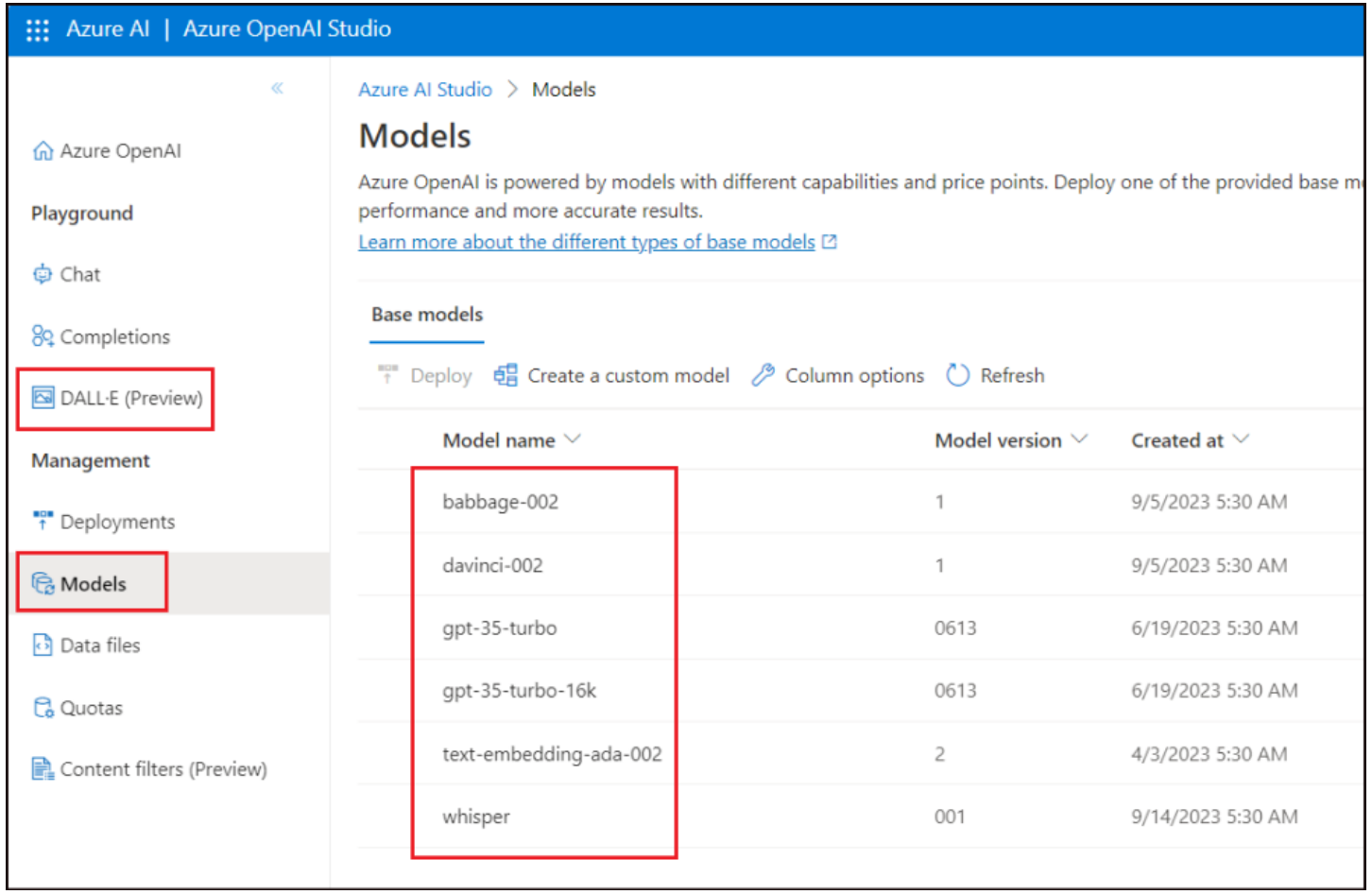
GPT3 base models

Embeddings series

DALL-E(Preview)

Whisper (Preview)

Fine-tuning models



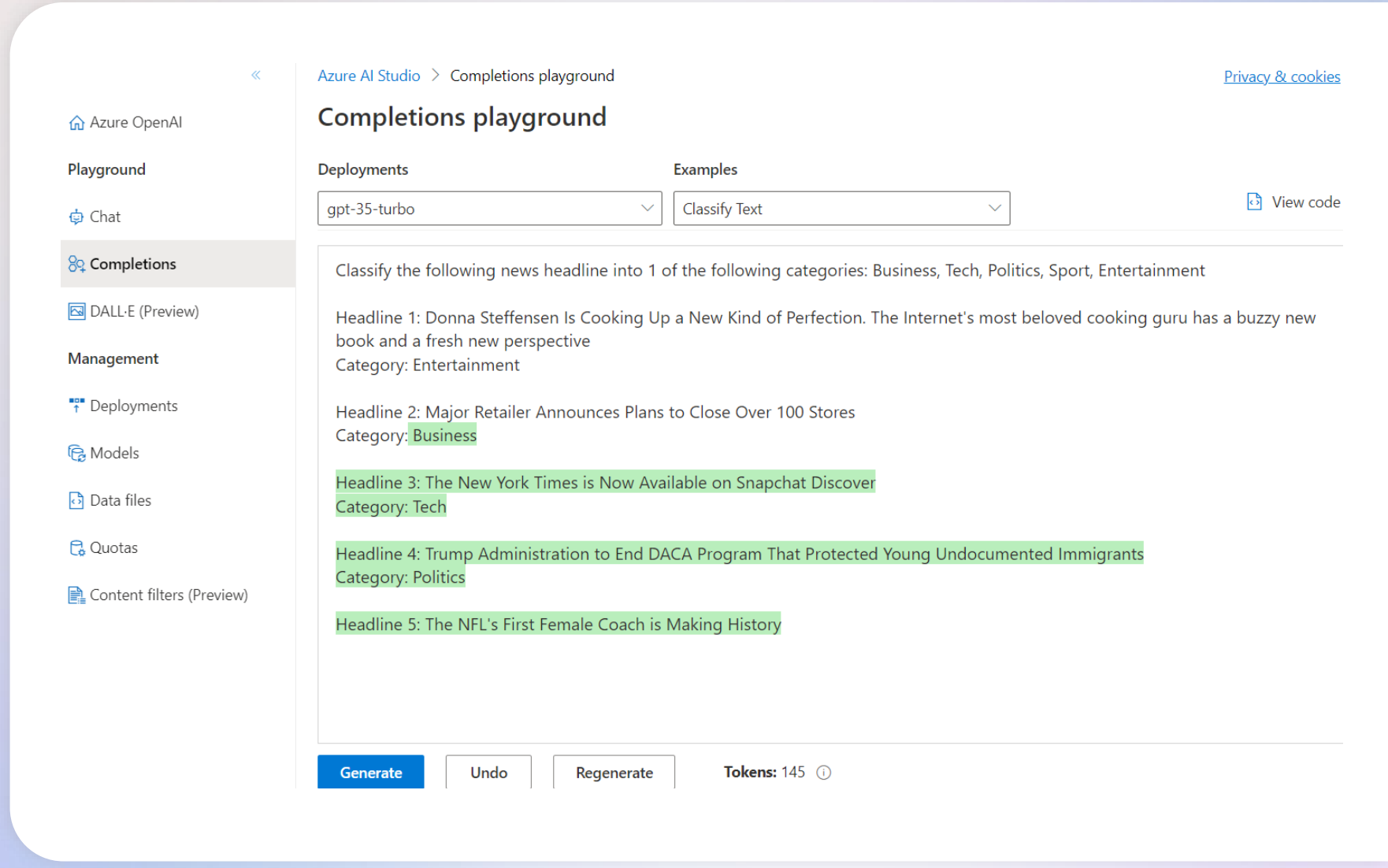
Existing Azure OpenAI customers can apply using this form: https://aka.ms/oai/get-gpt4

Prompts & Completions

Core component of the API service.

Text-in, text-out interface.

Provide an input **prompt**, and the model will generate a text **completion**.



Prompt types

Task type	Prompt example	Completion example
Classifying content	Tweet: I enjoyed the trip. Sentiment:	Positive
Generating new content	List ways of traveling	1. Bike 2. Car ...
Holding a conversation	A friendly AI assistant	
Transformation (translation and symbol conversion)	English: Hello French:	bonjour
Summarizing content	Provide a summary of the content {text}	The content shares methods of machine learning.
Picking up where you left off	One way to grow tomatoes	is to plant seeds.
Giving factual responses	How many moons does Earth have?	One

Text tokens

Tokens:

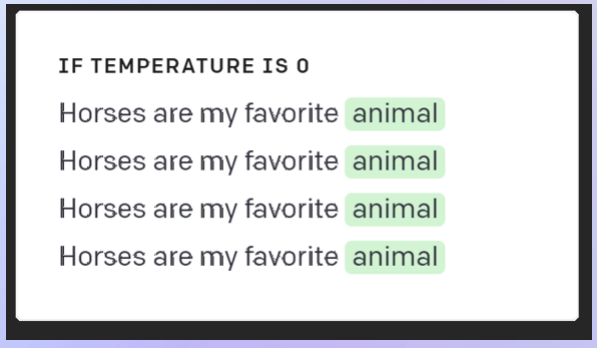
I have an orange cat named Butterscotch.

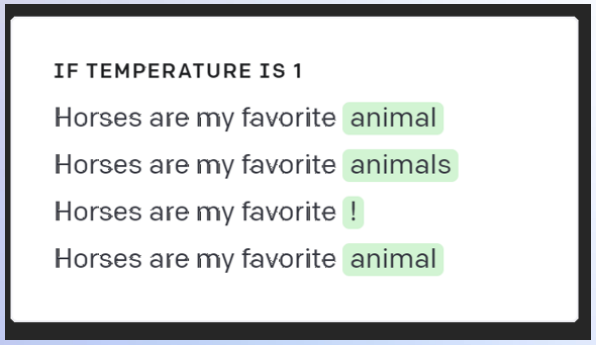
I have an orange cat named Butterscotch.

Horses are my favorite

animal	49.65%
animals	42.58%
\n	3.49%
!	0.91%

Probabilities:





Prompt Instruction

Suggest three names for an animal that is a superhero.

Animal: Cat
Names: Captain Sharpclaw, Agent Fluffball, The Incredible Feline
Animal: Dog
Names: Ruff the Protector, Wonder Canine, Sir Barks-a-Lot
Animal: Horse
Names:

Completion Temperature 0 (always the same)

Mighty Equine, The Great Galloper, Thunderhoof

Completion Temperature 1 (often different)

Blaze the Miracle Mare, Pegasus the Winged Warrior, Secretariat the Superhorse

Completion Temperature 1 (often different)

Blaze of Glory, Sterling Silver, Thunderbolt

Image tokens

Cost depends on

- Size
- Detail setting
  - Low resolution mode
  - High resolution mode

Low resolution mode	High resolution mode
Fewer input tokens	Uses more tokens
Cost 85 tokens each	Token cost is calculated by a series of scaling steps
4096 x 8192 image – Cost is a fixed 85 tokens	2048 x 4096 image – Cost is 1105 tokens after calculation

Azure OpenAI Service models

Models are grouped by family and capability

Model family	Description
GPT-4	A set of models that improve on GPT-3.5 and can understand generate natural language and code.
GPT-3.5	A set of models that improve on GPT-3 and can understand generate natural language and code.
GPT3 base models	Babbage-002 and Davinci-002 are GPT3 base models, intended for completion use cases.
Embeddings	A set of models that can convert text into numerical vector form to facilitate text similarity.
DALL-E(Preview)	A series of models that can generate original images from natural language.
Whisper(Preview)	A series of models in preview that can transcribe and translate speech to text.
Fine-tuning models	Let the customers tailor the Azure OpenAI models to the personal datasets.

Azure OpenAI Models

Models

Text, code and Vision	GPT-4 Turbo with Vision
Text and code	GPT-4 and GPT-4 Turbo Preview GPT-3.5, gpt-35-turbo and gpt-35-turbo-instruct, Babbage-002, Davinci-002
Image	DALL-E (Preview)
Voice	Whisper (Preview)
Embeddings	Embeddings
Fine tuning	GPT-3.5-Turbo (0613) Babbage-002 Davinci-002

Multi modality

Multimodal AI systems can analyze, synthesize, and generate across text, images.

Text and Code Models

Introduction to Text and Code models

GPT-3.5 models

Can understand and generate natural language and code

GPT-4 models

Can solve difficult problems with greater accuracy

GPT Model Use Cases

GPT-3.5

Large, pretrained Language Models that use deep learning to generate content

Use cases:

- Generating natural language for chatbots and virtual assistants with awareness of the previous history of chat
- Power chatbots that can handle customer inquiries, provide assistance, and converse but doesn't have memory of conversations
- Automatically summarize lengthy texts
- Assist writers by suggesting synonyms, correcting grammar and spelling errors, and even generating entire sentences or paragraphs
- Help researchers by quickly processing large amounts of data and generating insights, summaries, and visualizations to aid in analysis
- Generate good quality code based on natural language

GPT-4

Built-in responsible AI to detect and mitigate harmful use

Use cases:

- Generating and understanding natural language for customer service interactions, chatbots, and virtual assistants – doesn't have memory of conversations
- Generating high-quality code for programming languages based on natural language input.
- Providing accurate translations between languages
- Improving text summarization and content generation
- Provides for multi-modal interaction (text and images)
- Substantial reduction in Hallucinations
- Consistency between different runs is high

Text and Code models- Key Terms

Term	Definition

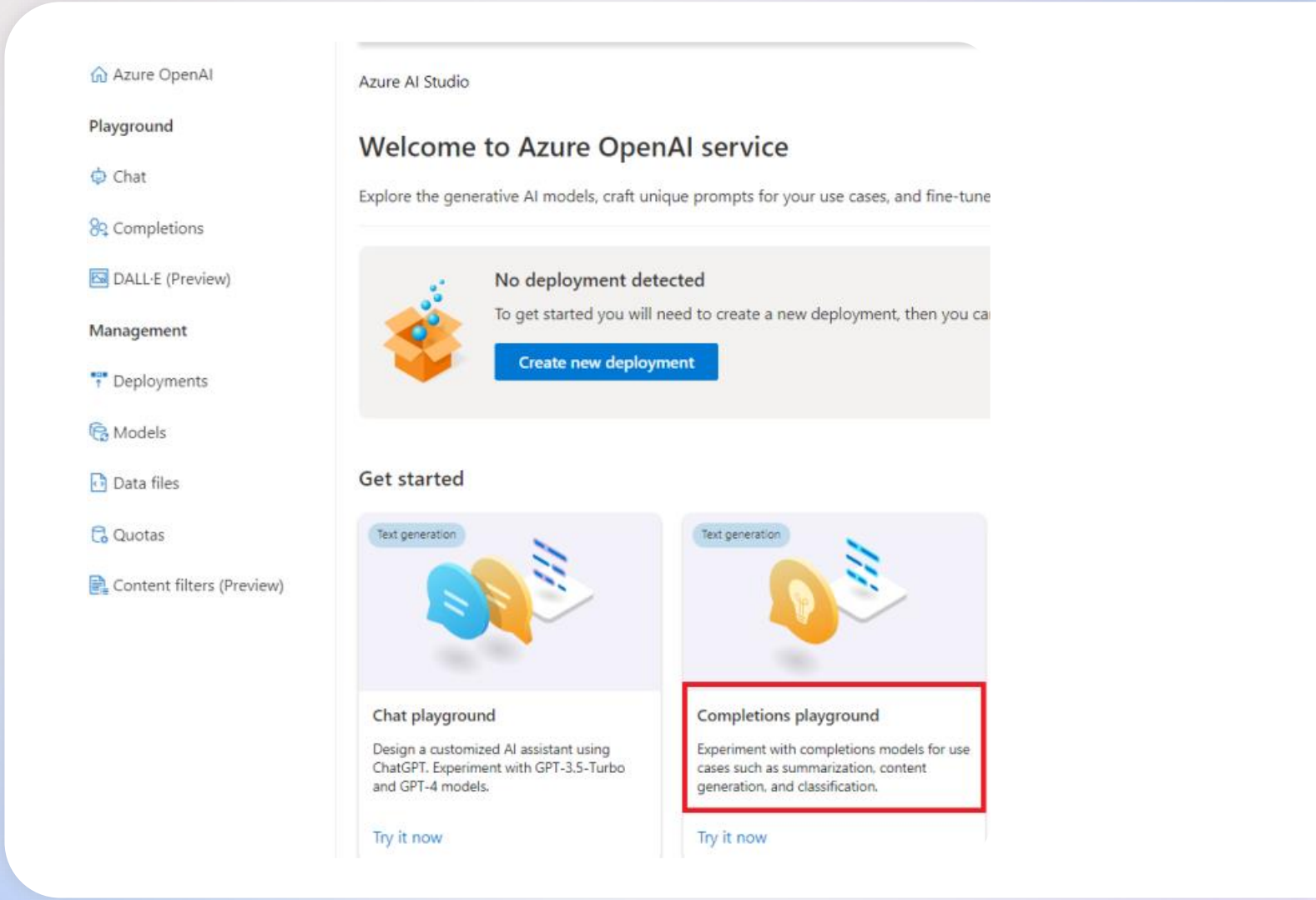
Prompt	The text you send to the service in the API call. This text is then input into the model. For example, one might input the following prompt:
	
	`Convert the questions to a command:`
	`Q: Ask Constance if we need some bread`
	`A: send-msg 'find constance' Do we need some bread?`
	`Q: Send a message to Greg to figure out if things are ready for Wednesday.`
	`A:`

Completion or Generation	The text Azure OpenAI outputs in response. For example, the service may respond with the following answer to the above prompt: `send-msg 'find greg' figure out if things are ready for Wednesday.`
Token	Azure OpenAI processes text by breaking it down into tokens. Tokens can be words or just chunks of characters. For example, the word `hamburger` gets broken up into the tokens `ham`, `bur` and `ger`, while a short and common word like pear is a single token. Many tokens start with a whitespace, for example `hello` and `bye.`

GPT-3.5 Turbo Instruct overview

Can understand and generate natural language or code.

This model is available to use with the Completions API.

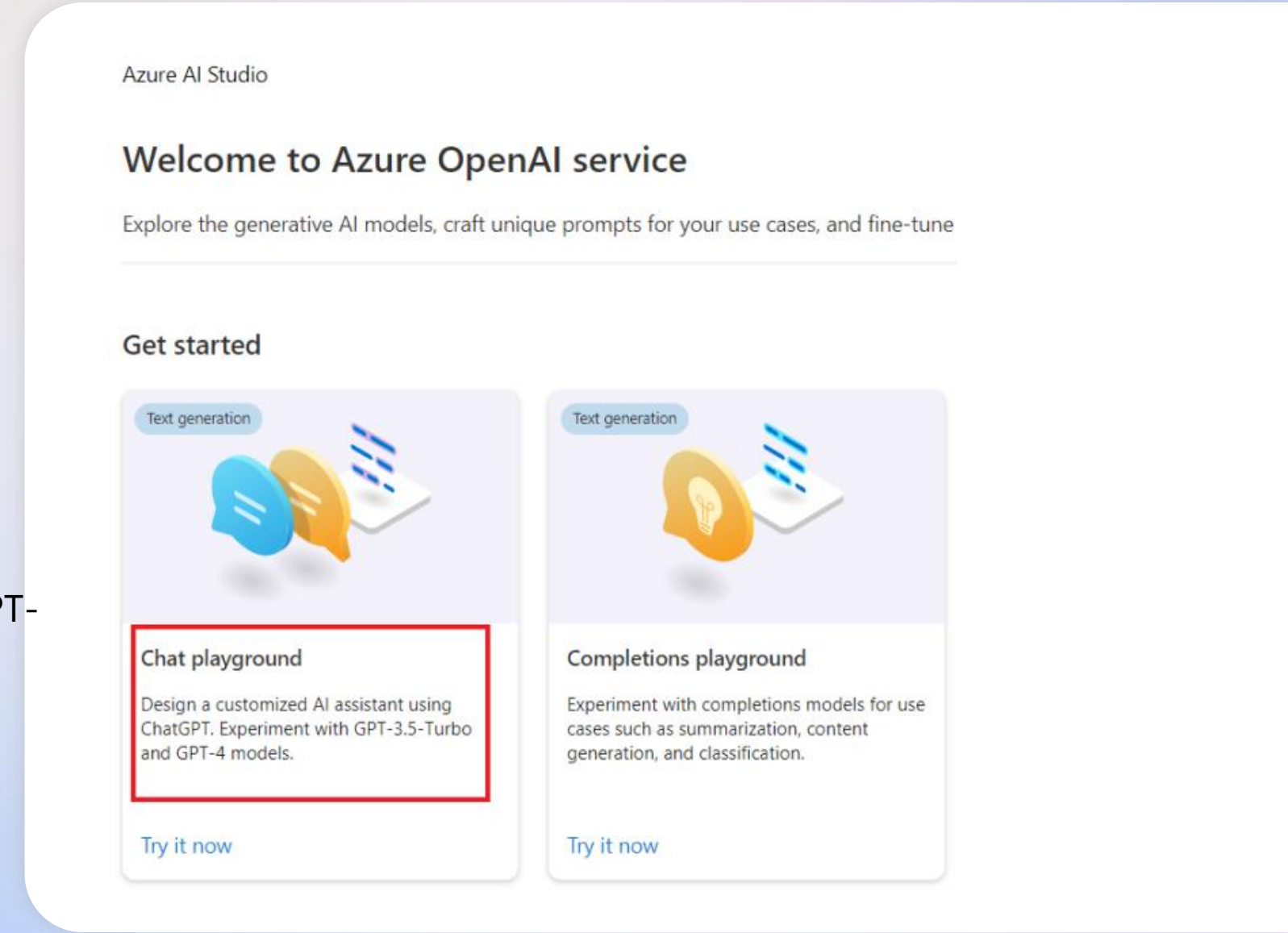


GPT3.5 overview

GPT-3.5 models can understand and generate natural language or code.

GPT-3.5 Turbo, has been optimized for chat and works well for traditional completions tasks as well.

Use the Chat Completions API to use GPT-3.5 Turbo.



GPT-3.5 models

Model ID	Max Request (tokens)
gpt-35-turbo(0301)	4,096
gpt-35-turbo (0613)	4,096
gpt-35-turbo-16k (0613)	16,384
gpt-35-turbo-instruct (0914)	4097
gpt-35-turbo (1106)	Input: 16,385 Output: 4,096

GPT-4 Overview

A set of models that improve on GPT-3.5 and can understand as well as generate natural language and code.

A large multimodal model (accepting text inputs, emitting text outputs) that,

Exhibits human-level performance on various professional and academic benchmarks.

Apply here for GPT-4 access(for already onboarded customers): https://aka.ms/oai/get-gpt4

GPT-4 is more reliable, creative, and able to handle much more nuanced instructions than GPT-3.5.

Can do everything that GPT 3.5 can do

Visual Question and Answering (VQA)

Steerability

context length of 8,192 tokens

GPT-4 models

Model ID	Max Request (tokens)
gpt-4 (0314)	8,192
gpt-4-32k (0314)	32,768
gpt-4 (0613)	8,192
gpt-4-32k (0613)	32,768
gpt-4 (1106-preview)1 GPT-4 Turbo Preview	Input: 128,000 Output: 4096
gpt-4 (vision-preview)2 GPT-4 Turbo with Vision Preview	Input: 128,000 Output: 4096

Babbage-002 and Davinci-002

Intended for completion use cases

Generate natural language or code

Not trained for instruction following

Working with the Chat models

There are two different APIs available for working with these type of models.

Chat Completion API (Recommended)

A new API designed for working with Chat models (gpt-35-turbo and gpt-4). We recommend using this API moving forward.

In this API, you pass in a series of messages rather than the prompt as a raw string making it easier to manage the conversation.

```
messages = [
  {
    "role": "system",
    "content": "Assistant is a large language model trained by OpenAI."
  },
  {
    "role": "user",
    "content": "What is a garbanzo bean?"
  }
]
```

Completion API

The GPT-35-Turbo model also works with the existing completions API. In this API, you'll format your prompt as a string and need to use the special tokens from Chat Markup Language.

We generally recommend using the Chat API instead because the special tokens may get updated in future version of gpt-35-turbo meaning you would need to update your prompts before updating to newer versions of the model.

```
<|im_start|>system
Assistant is a large language model trained by OpenAI.
<|im_end|>
<|im_start|>user
What is a garbanzo bean?
<|im_end|>
<|im_start|>assistant
```

Working with the Chat models

Previous GPT-3 models

Previous models were text-in and text-out

(i.e., they accepted a prompt string and returned a completion to append to the prompt).

Answer questions from the context below.

Context:
A neutron star is the collapsed core of a massive supergiant star, which had a total mass of between 10 and 25 solar masses, possibly more if the star was especially metal-rich.

Q: What is a neutron star?
A:

GPT-35-Turbo and GPT-4

The GPT-35-Turbo and GPT-4 models are conversation-in and message-out.

(i.e., it expects a prompt string that is formatted in a specific chat-like transcript format and returns a completion that represents a model-written message in the chat)

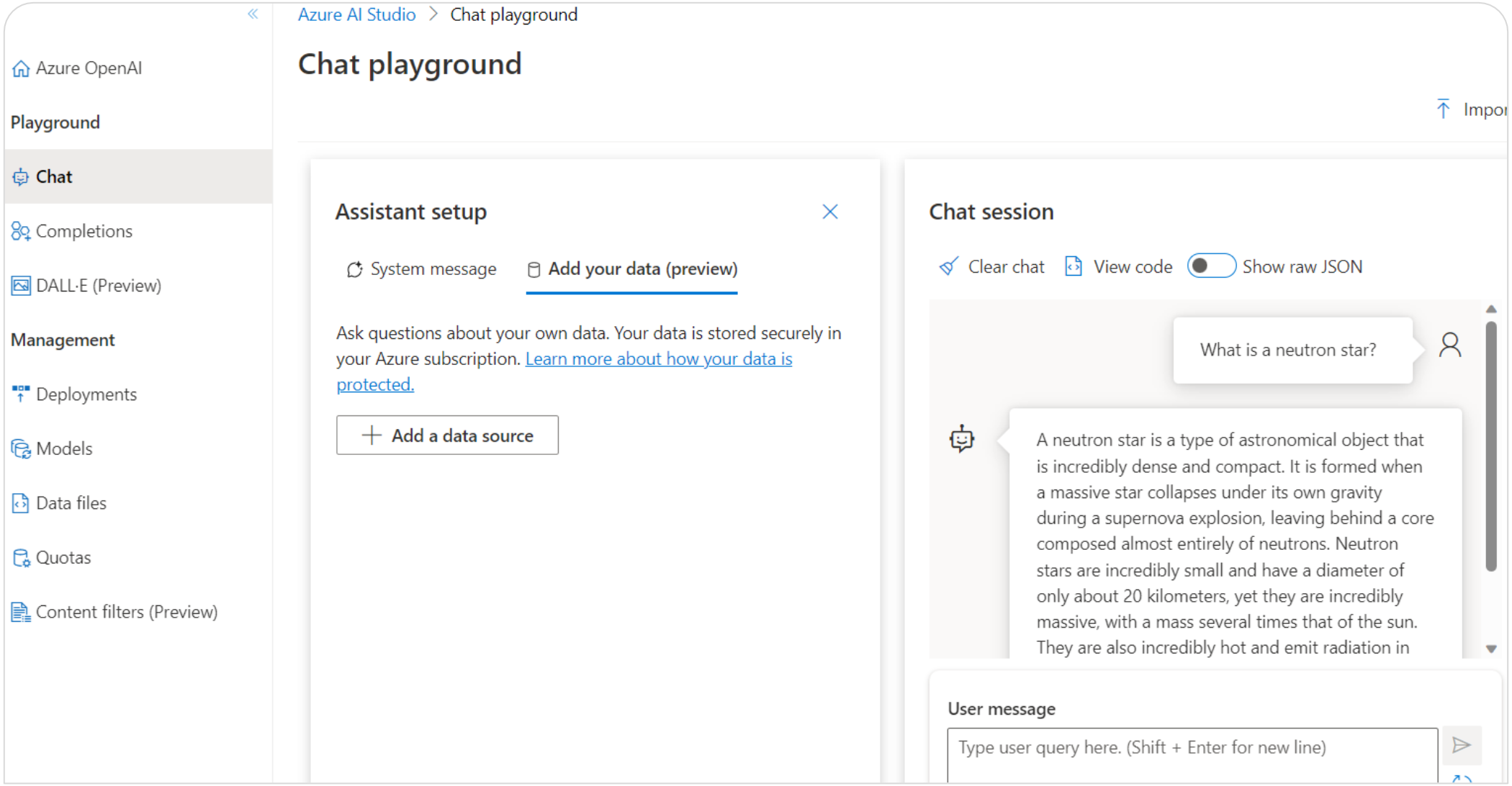
[System Message]
Assistant is an AI Chatbot designed to answer questions from the context provided below.

Context:
A neutron star is the collapsed core of a massive supergiant star, which had a total mass of between 10 and 25 solar masses, possibly more if the star was especially metal-rich.

[User]
What is a neutron star?

[Assistant]

An example in Chat playground



Understanding the prompt format

The system message

The system message is included at the beginning of the prompt and is used to prime the model and you can include a variety of information in the system message including:

- A brief description of the assistant
- The personality of the assistant
- Instructions for the assistant
- Data or information needed for the model

User and assistant messages

After the system message, you can include a series of messages between the *user* and the *assistant*. You denote who the message is from by setting the role to user or assistant.

```
{
    "role": "user",
    "content": "What is a garbanzo bean?"
}
```

Example prompt

{ "role": "system", "content": "You are an Xbox customer support agent whose primary goal is to help users with issues they are experiencing with their Xbox devices. You are friendly and concise. You only provide factual answers to queries, and do not provide answers that are not related to Xbox. },

{ "role": "user", "content": "Why won't my Xbox turn on?"},

{ "role": "assistant", "content": "There could be a few reasons why your Xbox isn't turning on...."},

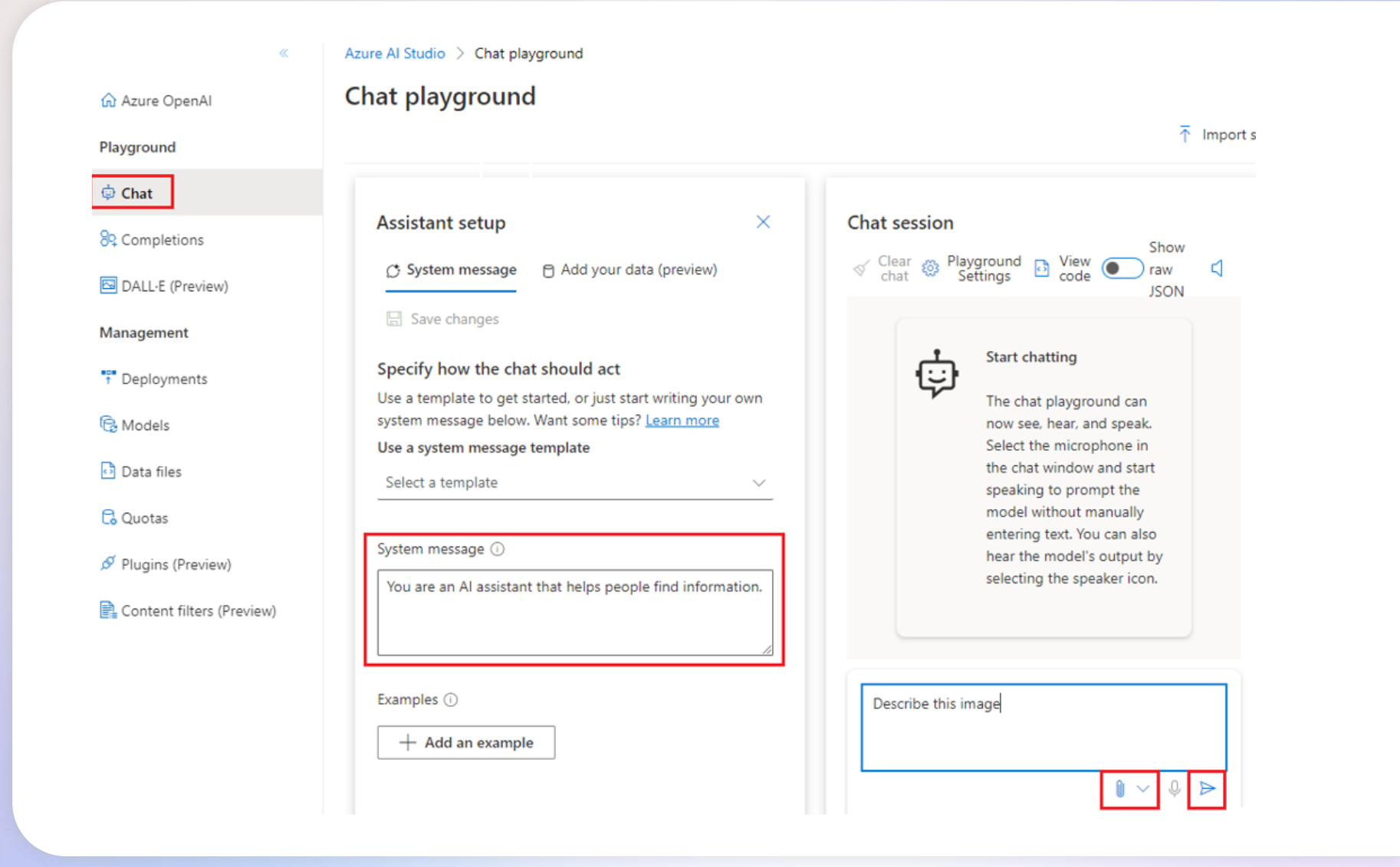
{ "role": "user", "content": "I confirmed the power cord is plugged in but it's still not working" }

GPT-4 Turbo with Vision

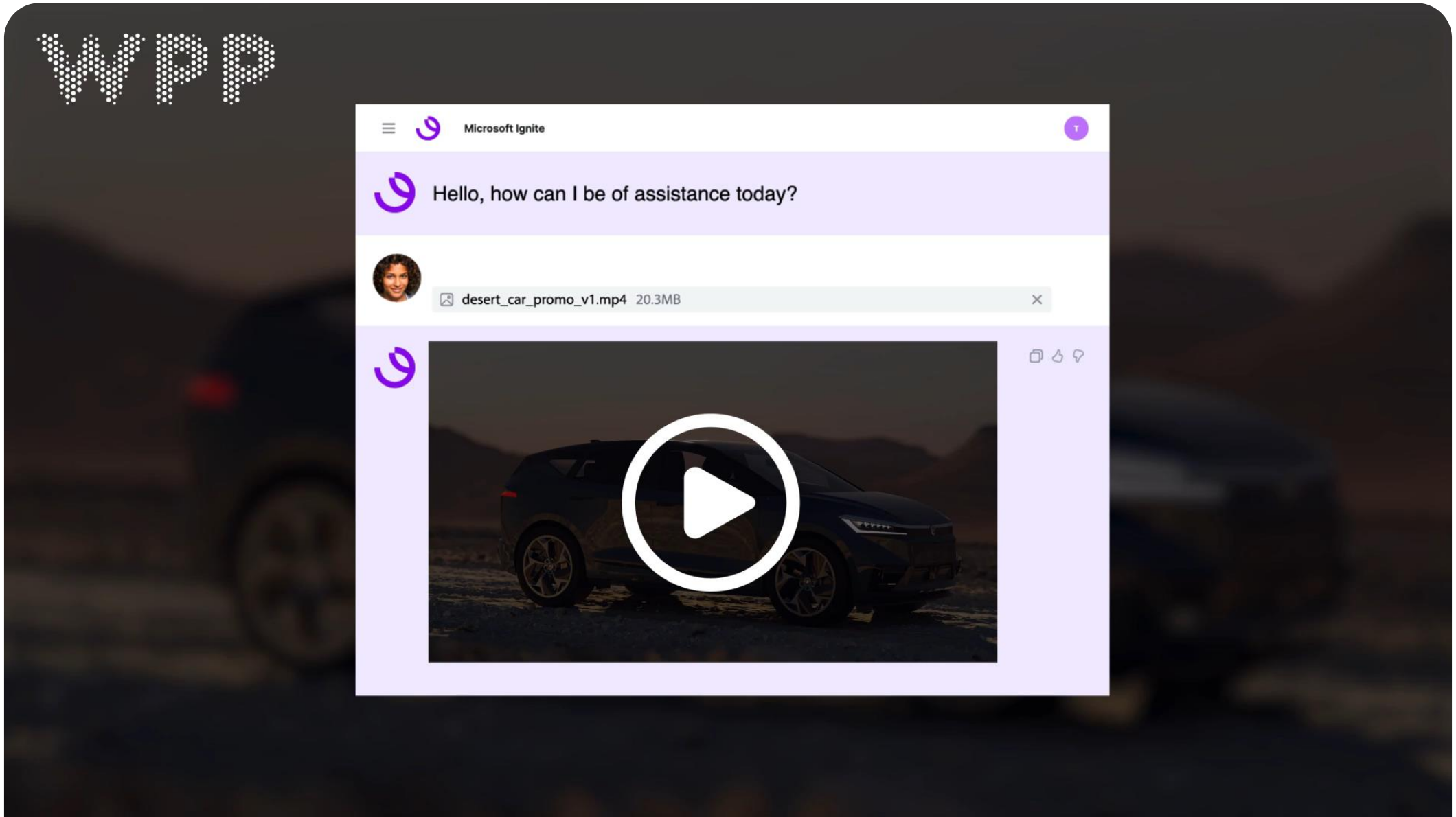
GPT-4 Turbo with Vision

A multimodal model that analyze images and provide textual responses to questions about them

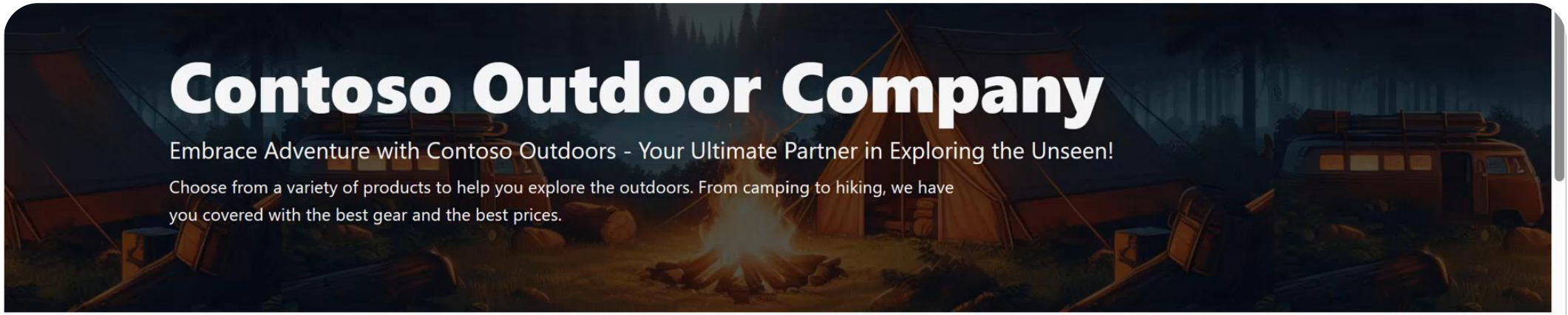
Incorporates both natural language processing and visual understanding.



GPT-4 Turbo with Vision + Azure AI Services



GPT-4 Turbo with Vision + Azure AI Services



Tents

UNFOLD ADVENTURE WITH CONTOSO'S RANGE OF TENTS - YOUR PORTABLE HOME UNDER THE stars. From intimate two-person camping shelters to family-sized canopies, our tents are your ticket to escaping the ordinary and embracing the great outdoors.

TrailMaster X4 Tent

Alpine Explorer Tent

SkyView 2-Person Tent

Combine Text & Image Inputs

User

Write a six-line rhyming poem. Starting each line with the letter “c".

The first three lines are around this image.

And the next three lines are around this image.

Vision AI

Calm and cool, the sea is blue,

Crashing waves, a scenic view,

Craggy rocks, a sight so true,

Colorful and warm, the sun is low,

Casting shadows, a golden glow,

Cozy and calm, a peaceful show.

Detailed Image Descriptions

Vision AI

A person with a striped shirt and an apron is holding a metal watering can and watering the plants in the greenhouse.

A large metal watering can with a long spout is being used by a person to water the green and leafy plants in the greenhouse.
A greenhouse full of green and leafy plants is being watered by a person with a metal watering can. The person is wearing a striped shirt and an apron.

A person is taking care of the plants in the greenhouse by watering them with a metal watering can. The person has a striped shirt and an apron on. The plants are green and leafy.

DALL-E 3(Preview)

DALL-E (Preview)

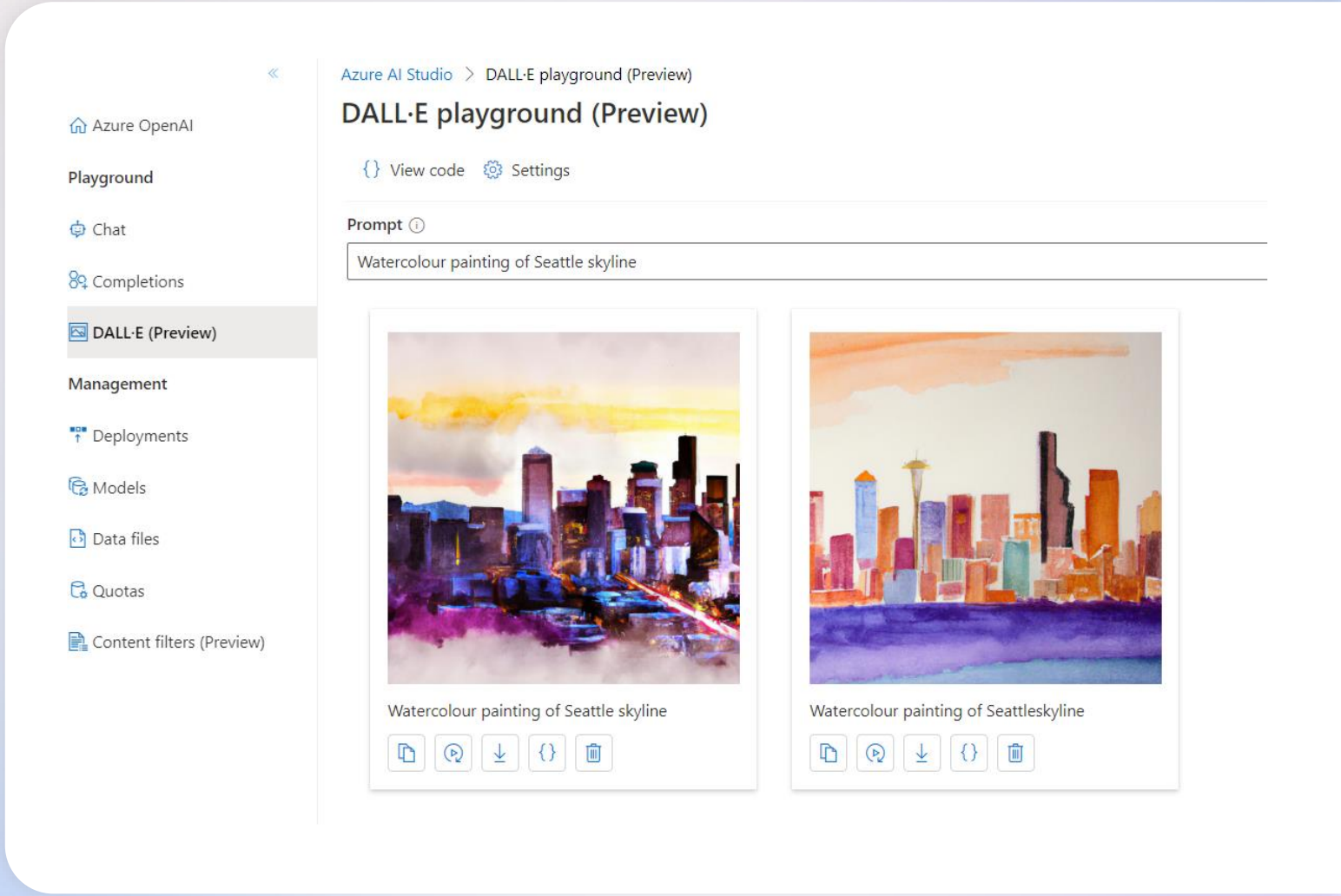
The model that works with images. Image capabilities include,

Image Generation

Editing an image

Creating variations of image

Dall-E 3 is now available in preview



DALL·E 3

Azure OpenAI Service

DALL-E 3 is an image generation model that allows you to generate images from text prompts

DALL·E 3 brings dramatic quality improvements

Enhanced image quality

"an image of a bear walking through the forest, close-up, photorealistic"

Ability to render more complex scenes

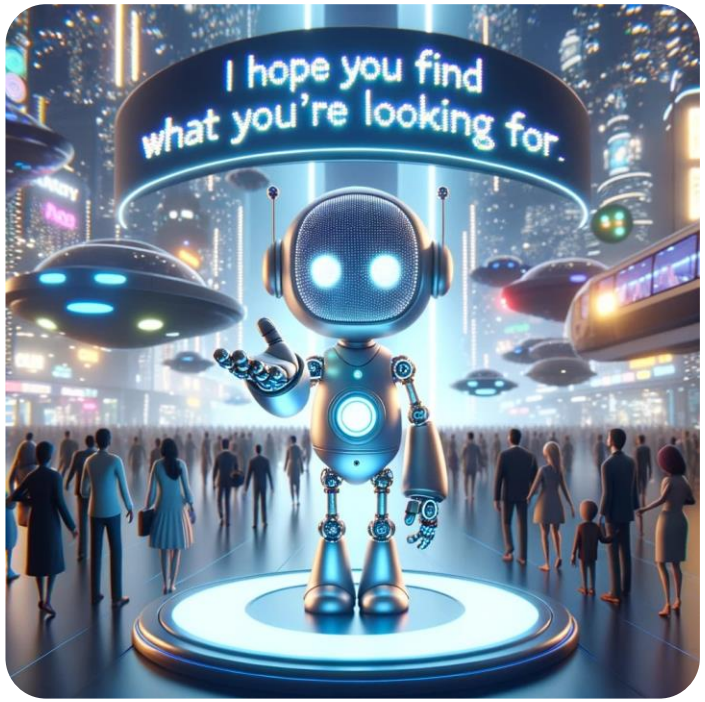
"An astronaut lounging on a chair made of clouds with the vivid night sky and earth brightly shining in the background. A large meteor streaks through the sky as the astronaut watches it fly by."

More realistic images of people and hands

"A group of business people meet in a conference room of a high-rise with a stunning city-scape in the background"

DALL·E 3 also offers new capabilities

Improved rendering of text in images



"a 3d render of a cute robot telling weary travelers "I hope you find what you're looking for", futuristic background"

Support for new aspect ratios

"A panorama of vast mountains with vivid mountain peaks, forests, and alpine lakes at sunrise"

Prompt rewriting in DALL·E 3

The DALL-E 3 API will include built-in prompt rewriting to enhance images, reduce bias, and increase natural variation

"A group of scientists in the lab"

"A painting of mount rainier, various styles"

Use Cases for DALL·E 3

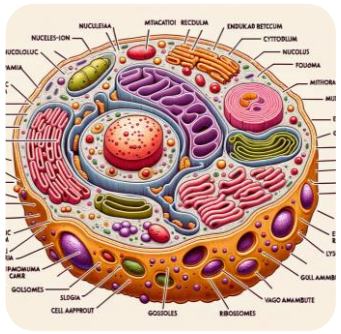
LOGO & BRANDING: QUICK CONCEPT GENERATION.

CREATIVE INSPIRATION: OVERCOME DESIGN BLOCKS.

CONTENT ILLUSTRATIONS: UNIQUE IMAGES FOR BLOGS/ARTICLES.

AD CAMPAIGNS: VISUALIZE MARKETING CONCEPTS.

PRODUCT VISUALIZATION: GAUGE INTEREST & FEEDBACK.



EDUCATION: CUSTOM IMAGERY FOR COURSES.

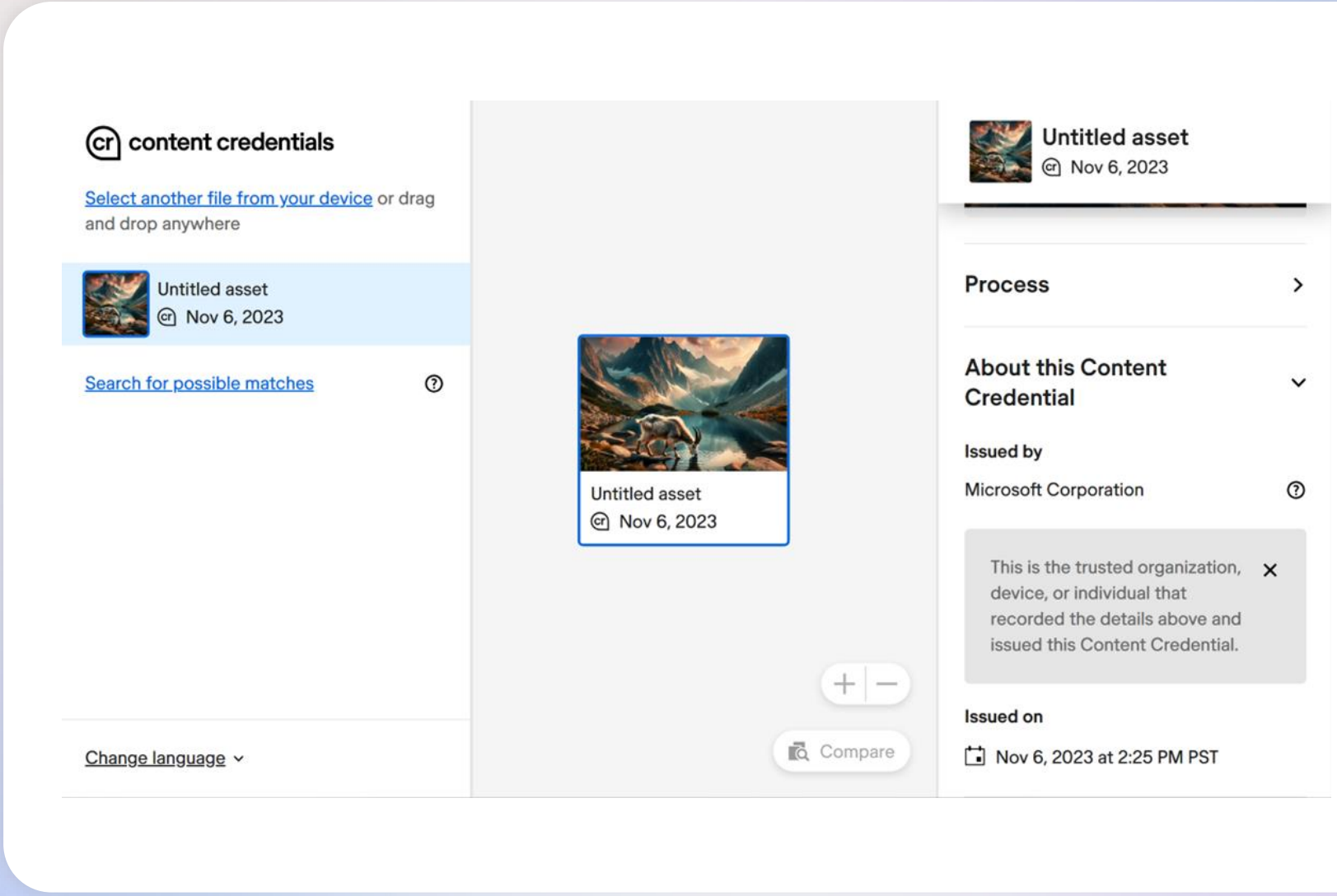
FASHION DESIGN: VISUALIZE CLOTHING PATTERNS.

GAMING: CHARACTER & ENVIRONMENT CONCEPTS.

Content Credentials

Provides information about the origin of an image generated by the DALL-E series models.

- Ways to check the Credential,
  - Content Credentials Verify webpage
  - Content Authenticity Initiative (CAI) JavaScript SDK



Provisioned throughput units(PTU)

What is Provisioned Throughput?

A new Azure OpenAI Service feature that lets customers **reserve model processing capacity** for running high-volume or latency-sensitive workloads

Reserved processing capacity provides consistent latency and throughput for workloads with consistent characteristics, such as prompt size, completion size, and number of concurrent API requests

Processing capacity is defined in units called “Provisioned Throughput Units” (PTUs) that are purchased on a monthly commitment

Once purchased, customers use PTUs to create provisioned Azure OpenAI deployments of GPT-35-turbo or GPT-4 models during the term of their commitment

Provisioned Throughput Key Values

Model processing capacity for your high-volume production workload

Predictable performance

Stable latency and throughput for uniform workloads

Reserved Processing Capacity

Processing capacity is available to meet demand

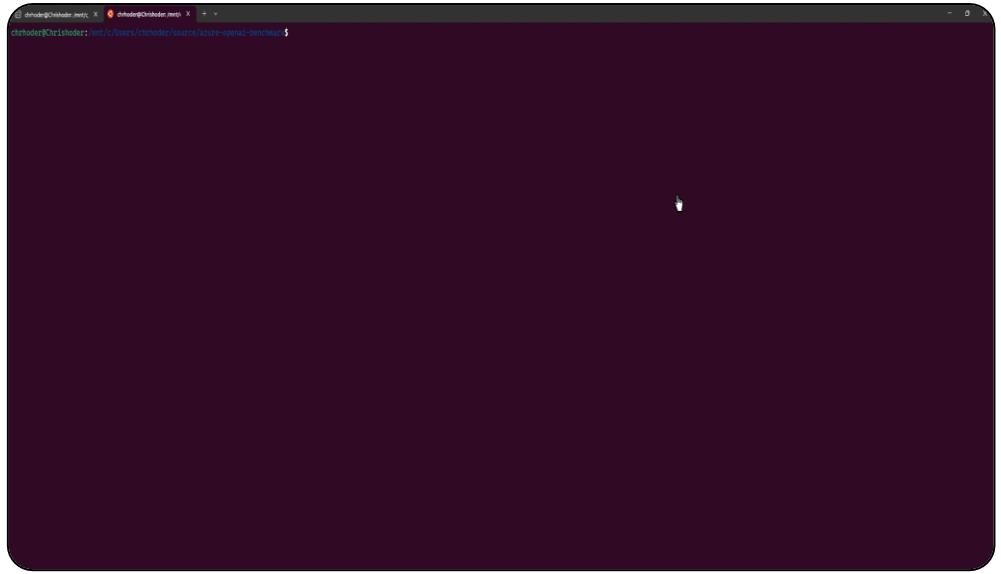
Cost Savings

High throughput workloads will likely see cost savings vs token-based consumption

Managing Provisioned Deployments

Performance Assesment

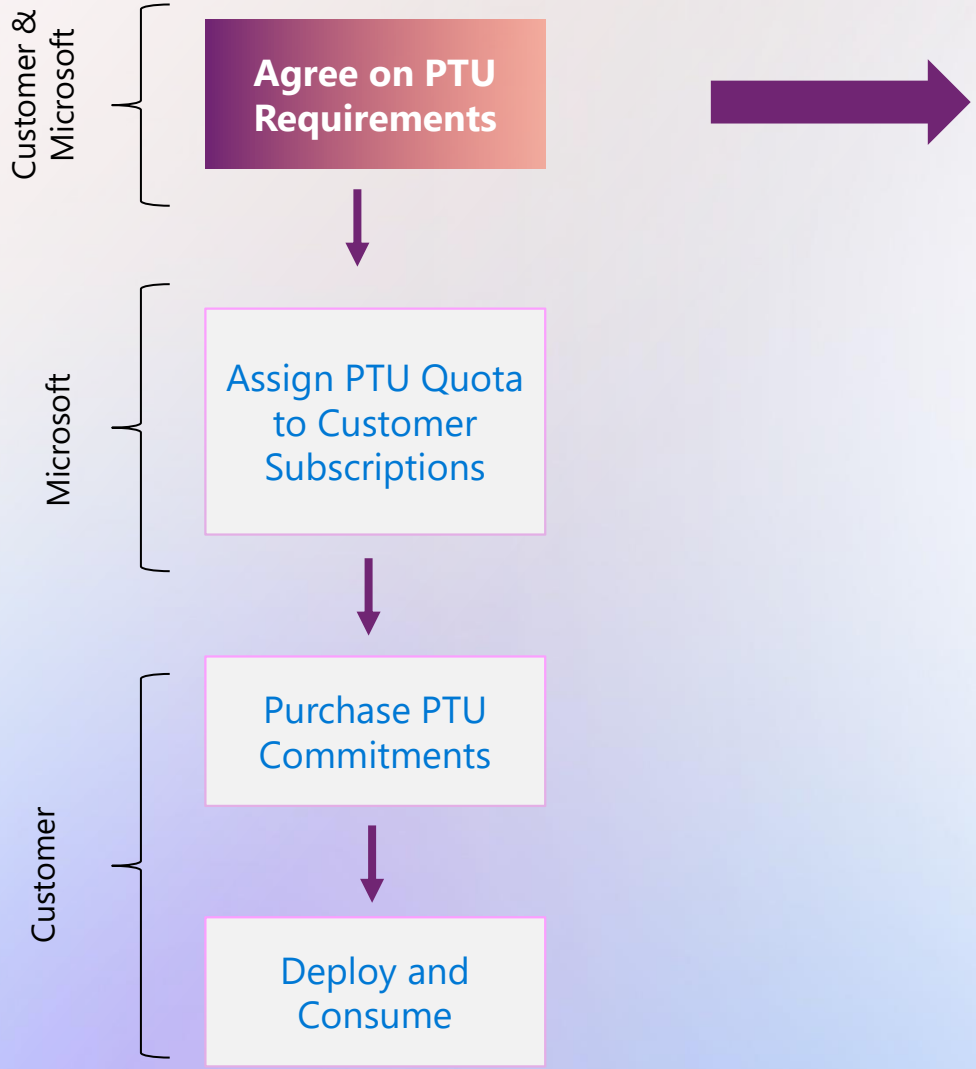
Azure OpenAI provides a Python toolset for assessing deployment performance across the range of prompt/generation token sizes and RPM



Production Monitoring

Deployment metrics are built into Azure Monitor, including utilization, latency, token and request counts

Provisioned Throughput Workflow



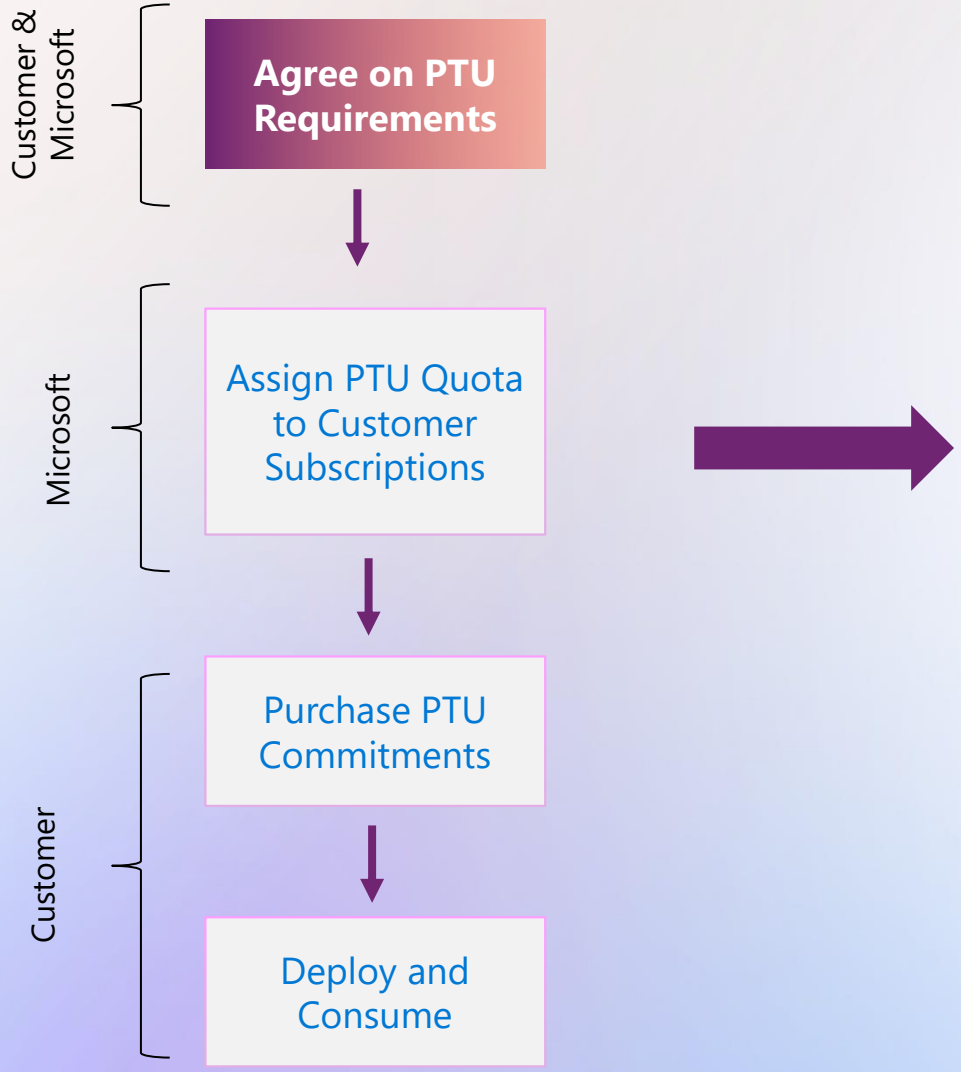
Customer works with their Microsoft account team to define PTU requirements by model and geographic region.

Customer uses online calculator to estimate required PTUs

Account team and product team collaborate to identify the Azure regions where capacity is available to meet customer requirements

Microsoft and Customer finalize agreement to purchase PTUs in the associated regions, including signing relevant contract amendments

Provisioned Throughput Workflow

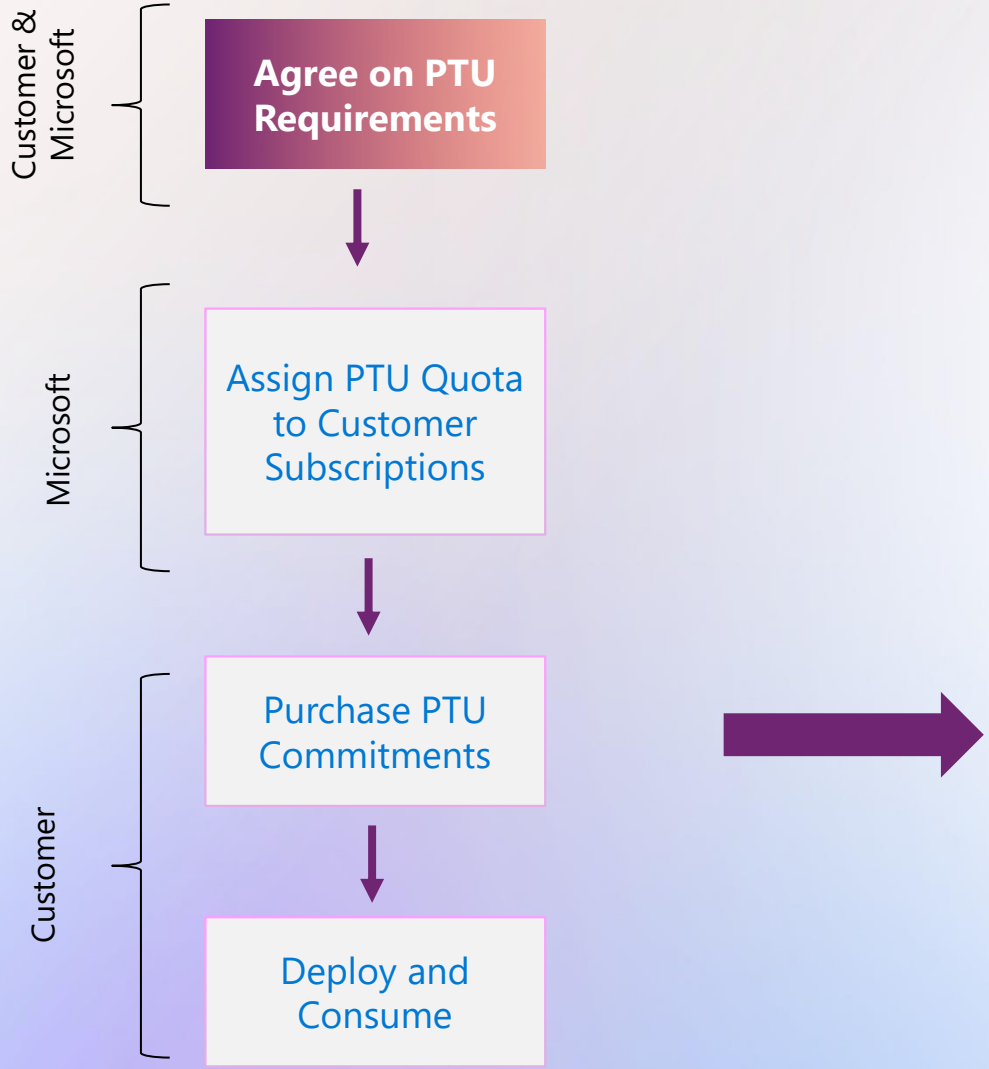


Microsoft product team reserves the capacity for the customer and assigns PTU quota to the subscription

PTU quota is assigned per-model type

The expectation is that the PTU quota will be purchased within 48 hours of assignment. Otherwise, PTUs may be reclaimed.

Provisioned Throughput Workflow



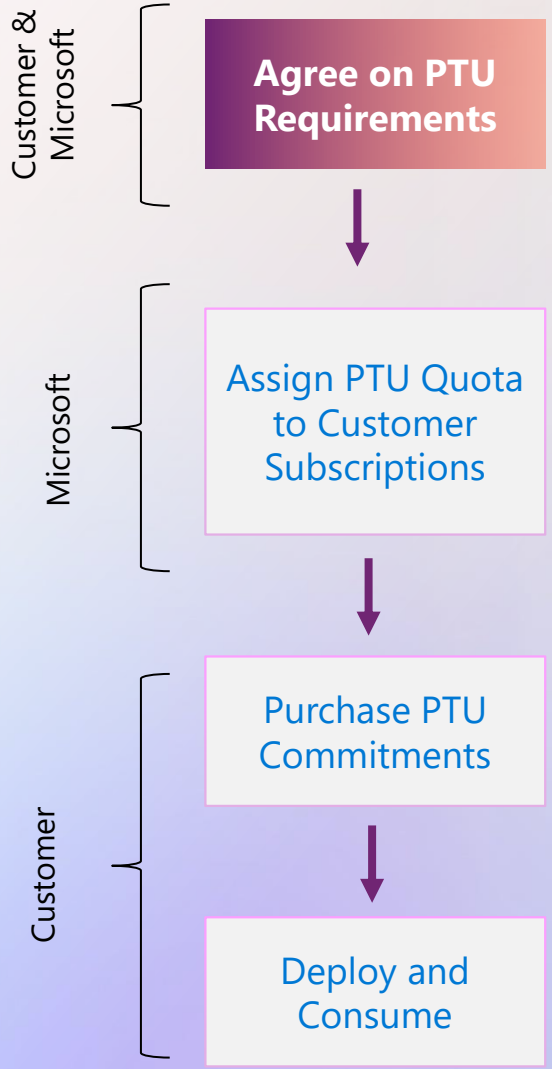
Customer uses Azure AI Studio to purchase the PTUs

Commitments are created at the resource level, not the subscription level, and cannot be transferred to other resources or subscriptions once created

Each resource can have its own commitment for a different number of PTUs (constrained by the sub quota limit)

Provisioned throughput deployments can only be created within resources that have commitments, and only up to the commitment amount

Provisioned Throughput Workflow



Provisioned deployments are created in the same way as standard, pay-as-you-go deployments

The difference is that the deployment type will be set to “Provisioned Managed”, and the desired number of PTUs will need to be set

Once a provisioned deployment is created, it can be scaled up or down in PTU increments.

Multiple provisioned deployments of the same or different model types can be created within the same resource

Assess performance using benchmarking tool

Monitor utilization via Azure Monitor

Using function calling with Azure OpenAI Service (Preview)

How to use function calling with Azure OpenAI Service (Preview)

Call the chat completions API with your functions and the user's input

Use the model's response to call your API or function

Call the chat completions API again, including the response from your function to get a final response

```
def run_conversation():
    # Step 1: send the conversation and available functions to the model
    messages = [{"role": "user", "content": "What's the weather like in San Francisco,
    tools = [
        {
            "type": "function",
            "function": {
                "name": "get_current_weather",
                "description": "Get the current weather in a given location",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": {
                            "type": "string",
                            "description": "The city and state, e.g. San Francisco, CA'
                        },
                        "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
                    },
                    "required": ["location"],
                },
            },
        }
    ]
```

Parallel function calling

Allows to perform multiple function calls together

```
    tool_calls = response_message.tool_calls
    # Step 2: check if the model wanted to call a function
    if tool_calls:
        # Step 3: call the function
        # Note: the JSON response may not always be valid; be sure to handle errors
        available_functions = {
            "get_current_weather": get_current_weather,
        }  # only one function in this example, but you can have multiple
        messages.append(response_message)  # extend conversation with assistant's re
        # Step 4: send the info for each function call and function response to the
        for tool_call in tool_calls:
            function_name = tool_call.function.name
            function_to_call = available_functions[function_name]
            function_args = json.loads(tool_call.function.arguments)
            function_response = function_to_call(
                location=function_args.get("location"),
                unit=function_args.get("unit"),
            )
            messages.append(
                {
                    "tool_call_id": tool_call.id,
                    "role": "tool",
                    "name": function_name,
                    "content": function_response,
                }
            )  # extend conversation with function response
        second_response = client.chat.completions.create(
            model="<REPLACE_WITH_YOUR_1106_MODEL_DEPLOYMENT_NAME>",
            messages=messages,
        )  # get a new response from the model where it can see the function respons
        return second_response
print(run_conversation())
```

Working with function calling

Defining functions

Managing the flow with functions

Prompt engineering with functions

```
response = openai.ChatCompletion.create(
    deployment_id="gpt-35-turbo-0613",
    messages=messages,
    functions=functions,
    function_call="auto",
)
response_message = response["choices"][0]["message"]

# Check if the model wants to call a function
if response_message.get("function_call"):

    # Call the function. The JSON response may not always be valid so make sure to h
    function_name = response_message["function_call"]["name"]

    available_functions = {
            "search_hotels": search_hotels,
    }
    function_to_call = available_functions[function_name]

    function_args = json.loads(response_message["function_call"]["arguments"])
    function_response = function_to_call(**function_args)

    # Add the assistant response and function response to the messages
    messages.append( # adding assistant response to messages
        {
            "role": response_message["role"],
            "function_call": {
                "name": function_name,
                "arguments": response_message["function_call"]["arguments"],
            }
```

Using function calling responsibly

- Validate Function Calls
- Use Trusted Data and Tools
- Follow the Principle of Least Privilege
- Consider Real-World Impact
- Implement User Confirmation Steps

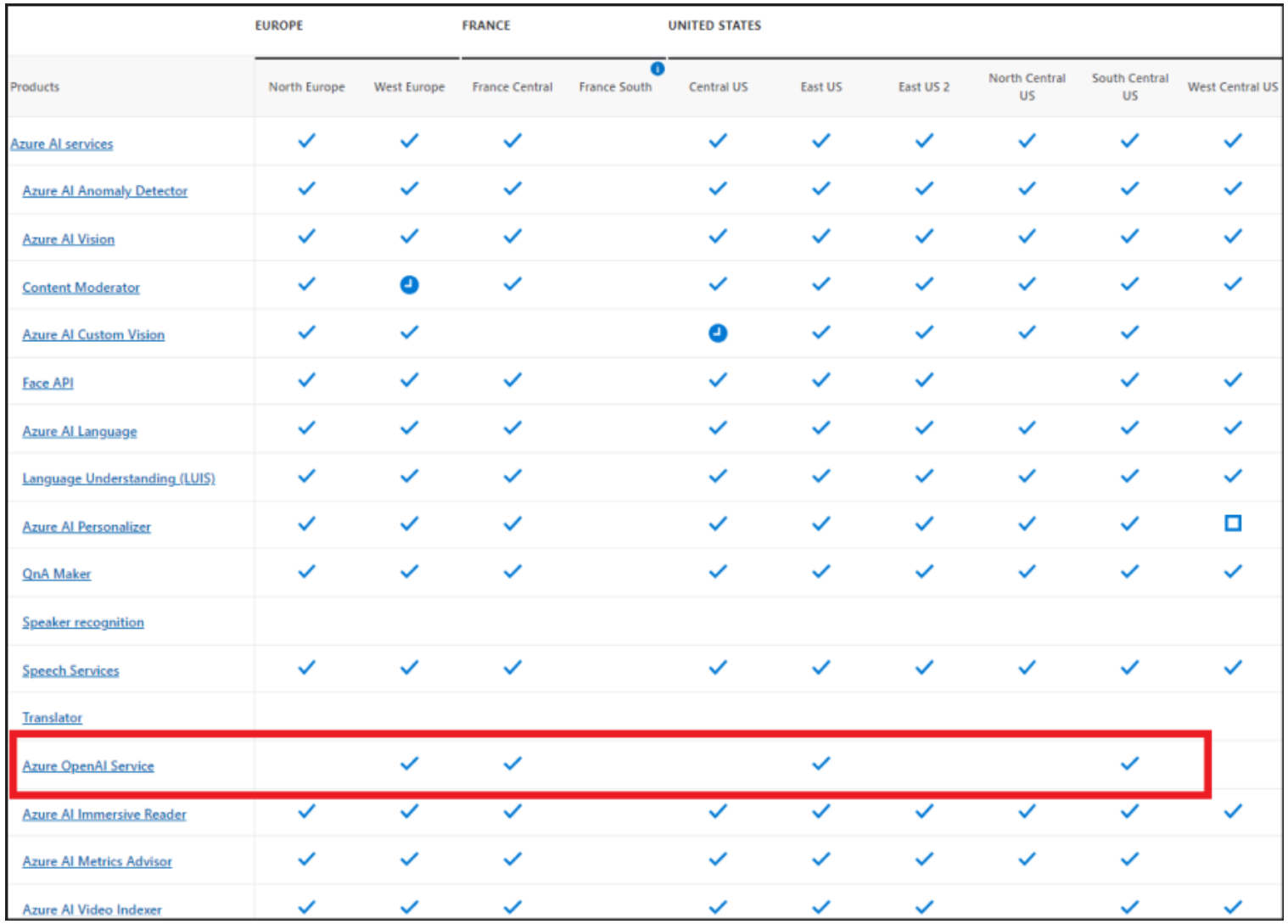
Availability and pricing

Region availability

Regions currently available at launch:

- Australia East
- Canada East
- East US
- East US 2
- France Central
- Japan East
- North Central US
- South Central US
- Sweden Central
- Switzerland North
- UK South
- West Europe

*Availability as of December 2023



	EUROPE		FRANCE		UNITED STATES					
Products	North Europe	West Europe	France Central	France South	Central US	East US	East US 2	North Central US	South Central US	West Central US
Azure AI services	✓	✓	✓		✓	✓	✓	✓	✓	✓
Azure AI Anomaly Detector	✓	✓	✓		✓	✓	✓	✓	✓	✓
Azure AI Vision	✓	✓	✓		✓	✓	✓	✓	✓	✓
Content Moderator	✓	◴	✓		✓	✓	✓	✓	✓	✓
Azure AI Custom Vision	✓	✓			◴	✓	✓	✓	✓	
Face API	✓	✓	✓		✓	✓	✓		✓	✓
Azure AI Language	✓	✓	✓		✓	✓	✓	✓	✓	✓
Language Understanding (LUIS)	✓	✓	✓		✓	✓	✓	✓	✓	✓
Azure AI Personalizer	✓	✓	✓		✓	✓	✓	✓	✓	□
QnA Maker	✓	✓	✓		✓	✓	✓	✓	✓	✓
Speaker recognition										
Speech Services	✓	✓	✓		✓	✓	✓	✓	✓	✓
Translator										
Azure OpenAI Service		✓	✓			✓			✓	
Azure AI Immersive Reader	✓	✓	✓		✓	✓	✓	✓	✓	✓
Azure AI Metrics Advisor	✓	✓	✓		✓	✓	✓	✓	✓	
Azure AI Video Indexer	✓	✓	✓		✓	✓	✓		✓	✓

Pricing

Pricing will be based on the pay-as-you-go consumption model

Price per unit for each model

Similar to other Azure Cognitive Services pricing models

General best practices to mitigate throttling during autoscaling

Implement retry logic in your application.

Avoid sharp changes in the workload.

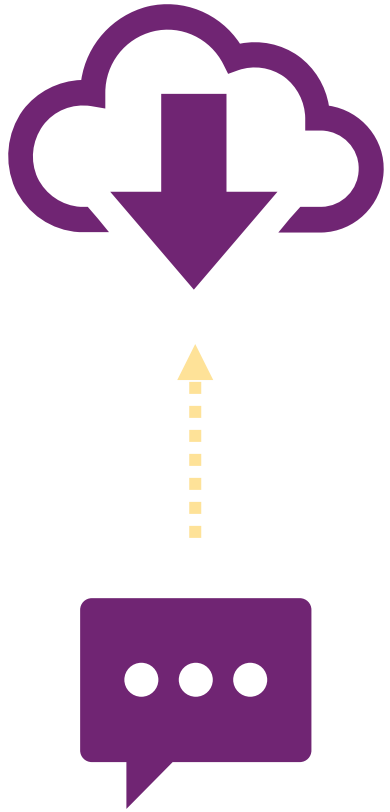
Test different load increase patterns. Create another OpenAI service resource and distribute the workload among them.

Apply prompt engineering with Azure OpenAI Service

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            | troglodytes    |
| 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

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

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

Knowledge check

1 **How can developers optimize the performance of Azure OpenAI models?**

- ❑ By using complex instructions that are difficult to understand
- ☑ By providing clear and descriptive instructions
- ❑ By using vague prompts

---

2 **What is the purpose of the system message in a prompt?**

- ☑ To give the model instructions, perspective, or other information helpful to guide its response
- ❑ To give the model a specific answer to generate
- ❑ To provide filler information to the model

---

3 **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

Generate code with Azure OpenAI Service

Use natural language prompts to generate code

- Use prompt engineering to describe what you want clearly
- Break down complex tasks into smaller pieces

Prompt:

`write a function for binary search in python`

```
def binary_search(array, target):
    low = 0
    high = len(array) - 1

    while low <= high:
        mid = (low + high) // 2
        if array[mid] == target:
            return mid
        elif array[mid] < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1
```

Change programming language and understand unknown code

Change programming language

```
Convert the following code to C#:
def print_squares(n):
    for i in range(1, n+1):
        print(i**2)
```

```
public void PrintSquares(int n)
{
    for (int i = 1; i <= n; i++)
    {
        Console.WriteLine(i * i);
    }
}
```

Understand unknown code

```
Could you explain what this code is doing?:
fn qzplv(n: i32, m: i32) -> i32 {
    if n == 0 {
        return m + 1;
    } else if m == 0 {
        return qzplv(n - 1, 1);
    } else {
        return qzplv(n - 1, qzplv(n, m - 1));
    }
}
```

```
Sure, I'd be happy to explain what this code is
doing!

This code is implementing the Ackermann
function,...
```

Complete partial code

- Writing your prompt with the most amount of information available will be the most accurate
- If enough code is given to determine the programming language, the model will use that language

Prompt:

```
complete the following function
"""
# calculate the average of the numbers
# in an array, but only if they're even
def
```

```
# calculate the average of the even
# numbers in an array
def calculate_even_average(numbers):
    total = 0
    count = 0
    for number in numbers:
        if number % 2 == 0:
            total += number
            count += 1
    if count == 0:
        return 0  # avoid division by zero
    else:
        return total / count
```

Write unit tests

Prompt:

```
write three unit tests for this function
"""
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

```
def test_binary_search():
    # Test case 1: Target element is in the middle of the
    # array
    arr1 = [1, 2, 3, 4, 5, 6]
    target1 = 4
    assert binary_search(arr1, target1) == 3

    # Test case 2: Target element is at the beginning of
    # the array
    arr2 = [1, 2, 3, 4, 5, 6]
    target2 = 1
    assert binary_search(arr2, target2) == 0

    # Test case 3: Target element is not in the array
    arr3 = [1, 2, 3, 4, 5, 6]
    target3 = 7
    assert binary_search(arr3, target3) == -1
```

Fix bugs in your code

- Help identify and fix bugs quickly
- Provide an explanation of the issues

Prompt:

```
This function isn’t working for me. Fix
the bugs in this function
"""
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        number = numbers[i]
        total += number
    average = total
    rerun averave
```

```
def calculate_average(numbers):
    total = 0
    for i in range(len(numbers)):
        number = numbers[i]
        total += number
    average = total / len(numbers)
    return average


There were a few issues with the original code:


1. The variable "average" was assigned the value of
"total" instead of the actual average.
2. There was a typo in the last line - "rerun" should be
"return".
3. The average was not actually calculated - the corrected
code divides the total by the length of
the "numbers" list to get the average.
```

Improve performance

- Identify where improvements can be made
- Models able to optimize and refactor

Prompt:

```
Can this function be more efficient?
"""
def sum_of_n(n):
    result = 0
    for i in range(1, n+1):
        result += i
    return result
```

```
We can improve this function it by using a mathematical
formula instead.

Here's the more efficient version:

def sum_of_n(n):
    return n * (n + 1) // 2
```

Exercise: Generate and improve code with Azure OpenAI Service

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

https://aka.ms/mslearn-openai-code-gen

Knowledge check

1 **What are some benefits of using Azure OpenAI to generate code?**

- [x] Increase in efficiency and productivity
- [ ] Increase in bugs and readability
- [ ] Increase in time spent coding

---

2 **What is the purpose of providing more context to the Azure OpenAI model when completing partial code?**

- [ ] Providing more context makes the model less accurate
- [ ] Providing more context doesn't affect the accuracy of the model
- [x] The more context provided to the model, the more accurate the response likely is

---

3 **What is an example of how Azure OpenAI models can change coding language?**

- [ ] Azure OpenAI models can only translate code from one language to another if the code is written in C#
- [x] If you have code in one language, but need it in another, Azure OpenAI can translate that for you in several languages
- [ ] Azure OpenAI models can only translate code from Python to C# or Java

Generate images with Azure OpenAI Service

What is DALL-E?

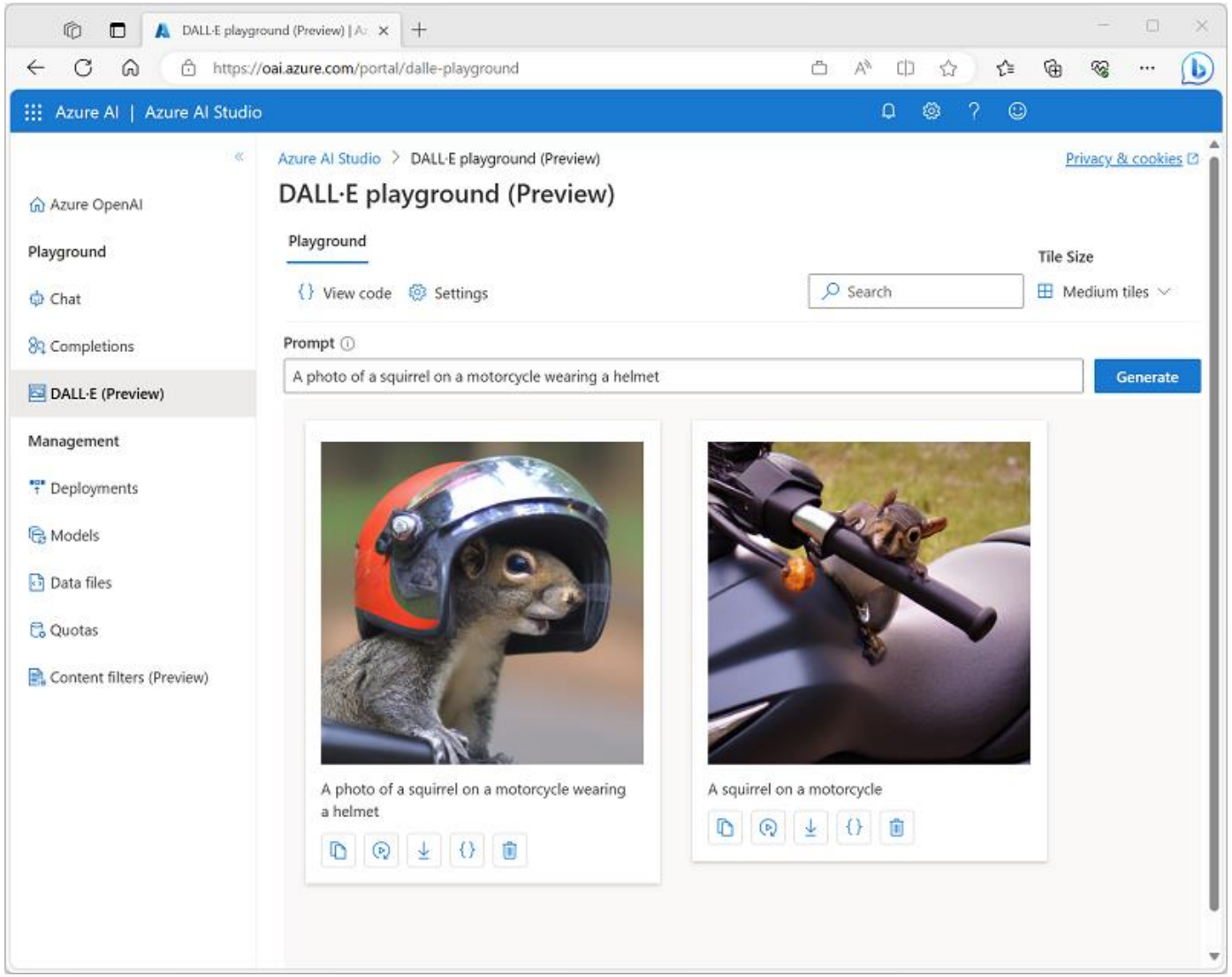
Generate images with a description

- Neural network based model for generating images
- Use natural language to describe what the image should be
- Specify content and style

Prompt:

```
a highland cow in a field on the coast of Scotland, digital art style
```

Using DALL-E in Azure OpenAI Studio

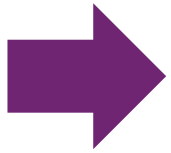


Using the Azure OpenAI REST API

DALL-E Endpoint

https://*endpoint*.openai.azure.com/openai/images/generations:submit?api-version=*api_version*

```
{
    "prompt": "A badger wearing a tuxedo",
    "n": 1,
    "size": "512x512"
}
```



```
{
    "created": 1685130482,
    "expires": 1685216887,
    "id": "088e4742-89e8-4c38-9833-c294a47059a3",
    "result":
    {
        "data":
        [
            {
                "url": "<URL_to_generated_image>"
            }
        ]
    },
    "status": "succeeded"
}
```

Using the Azure OpenAI SDKs for DALL-E

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

Code structure follows a similar pattern for both languages.

The response content comes in the form of an image URL, for both REST and SDKs. The image can be downloaded or viewed directly from that URL.

Pseudo code structure:

```
<include library>

<create client>

<define image generation prompt and
options>

<send request (can include options and
prompt in request)>

<extract response content>
```

Exercise - Generate images with a DALL-E model

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

https://aka.ms/mslearn-openai-dall-e

Knowledge check

1 **You want to use a model in Azure OpenAI to generate images. Which model should you use?**

- [x] DALL-E
- [ ] GPT-35-Turbo
- [ ] Text-Davinci

2 **Which playground in Azure OpenAI Studio should you use to explore image generation**

- [ ] Completions
- [x] DALL-E
- [ ] Chat

3 **In an API request to generate images, what does the n parameter indicate?**

- [ ] The description of the desired image
- [x] The number of images to be generated
- [ ] The size of the image to be generated