

Image Generation

- The Azure OpenAI service enables one to use large language models (LLMs) to generate content based on natural language prompts.
- One of these models is the DALL-E image generation model, which can create original graphical content based on natural language descriptions (prompts) of a desired image using deep learning methodologies.
- The ability to use AI to generate graphics has several applications, including the creation of illustrations or photorealistic images of articles or marketing collateral, generation of unique product or company logos, or any scenario where a desired image can be desired.

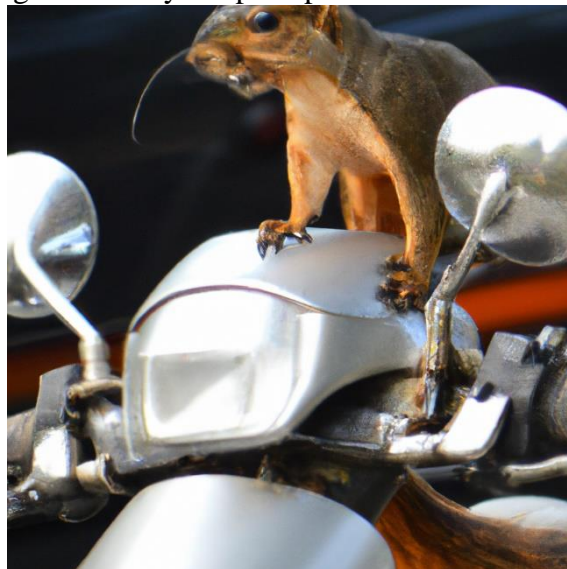
DALL-E

DALL-E is a neural network-based model that can generate graphical data from natural language input, exhibiting a broad understanding of visual and design trends. Put more simply, one can provide DALL-E with a description, and it can generate an appropriate image. It can generate imagery in multiple styles, including photorealistic imagery, paintings, and emoji. It can "manipulate and rearrange" objects in its images and can correctly place design elements in novel compositions without explicit instruction.

For example, submit the following natural language prompt:

A squirrel on a motorcycle

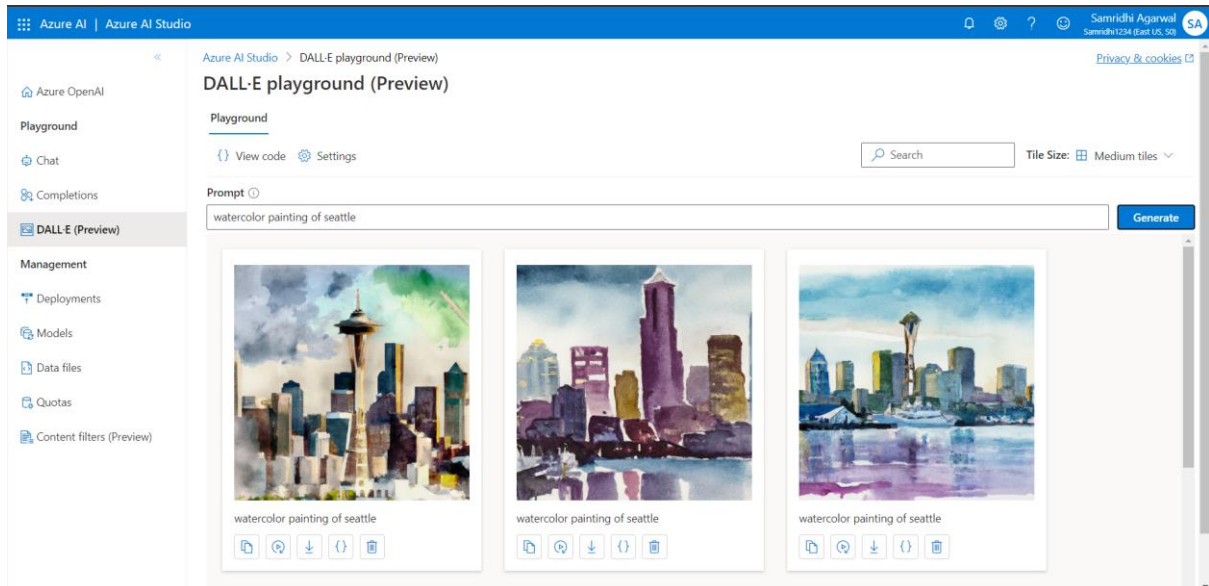
Graphical output generated by the prompt is such as following:



Note: These produced photos are original and not from a catalog. It is not a search engine for locating suitable photographs; it is an AI model that produces new images from its learned data.

DALL-E in Azure OpenAI Studio

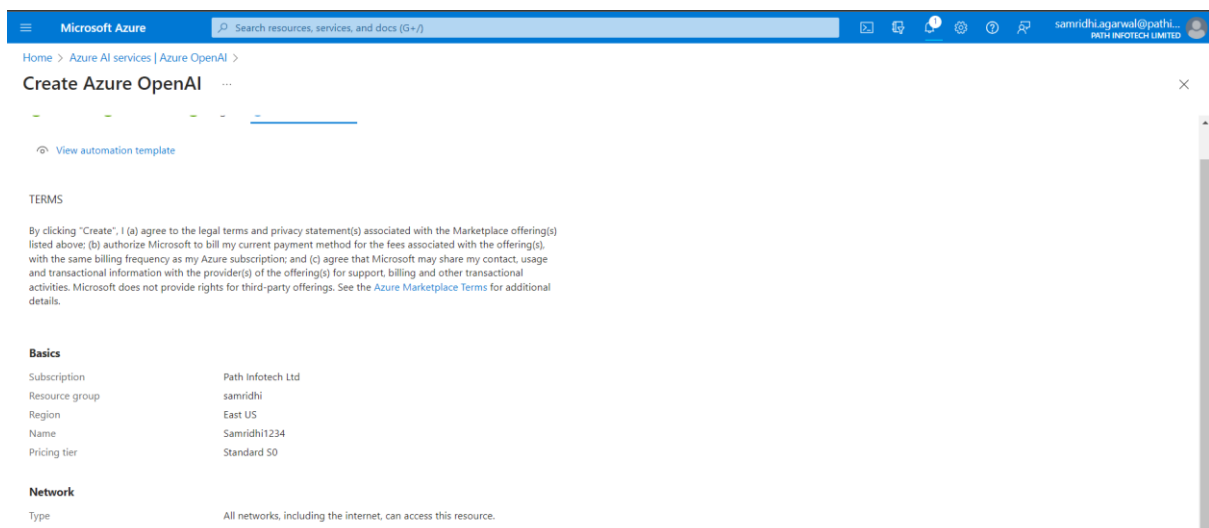
Experimenting with DALL-E, one can provision an Azure OpenAI Service resource in an Azure OpenAI Studio with subscription that has been approved for access to the service and use DALL-E playground to submit prompts and view the resulting generated images.



As it has been recently launched, DALL-E playground preview is available on Azure OpenAI Studio.

Generating images using DALL-E model

1. Create an Azure OpenAI resource:

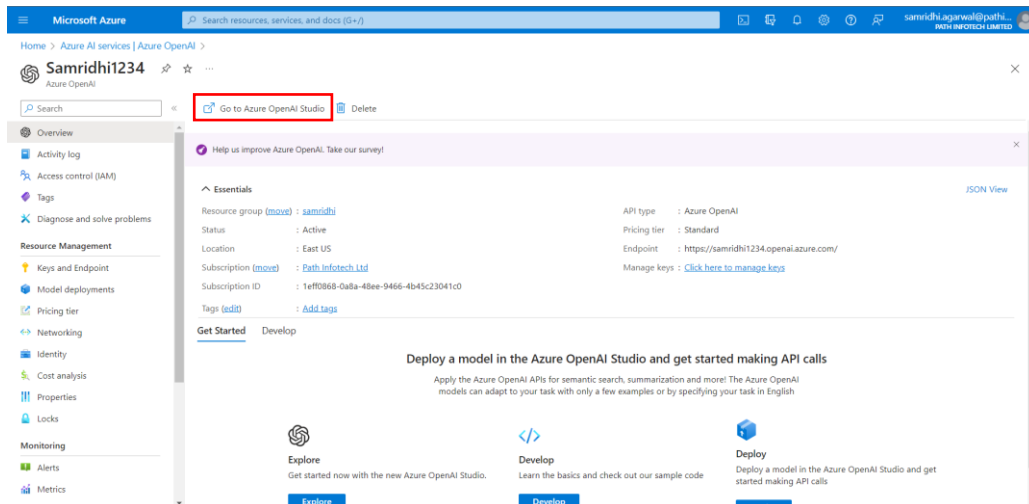


Wait for the deployment to complete. Then go to the Azure OpenAI resource in the Azure portal.

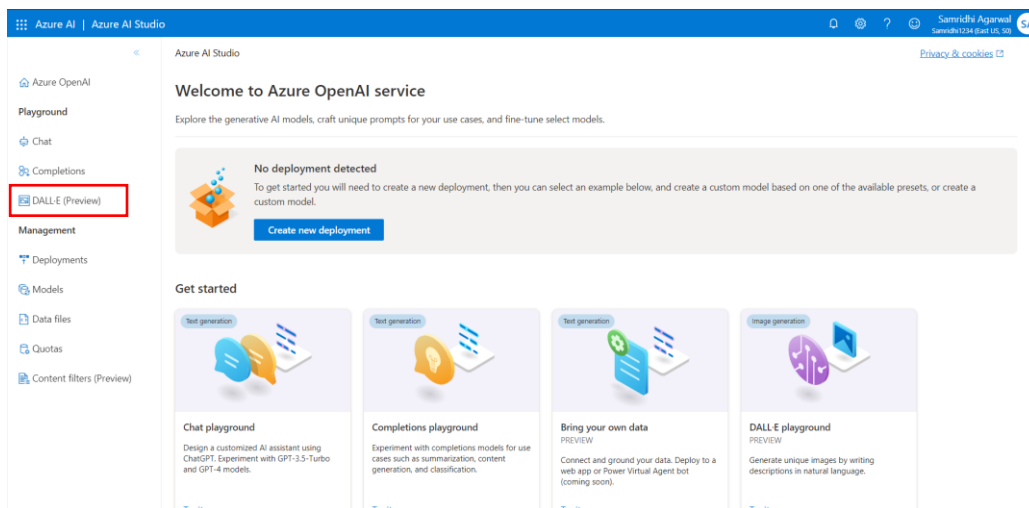
Image generation in DALL-E playground

Azure OpenAI Studio's DAAL-E playground allows picture generating experiments.

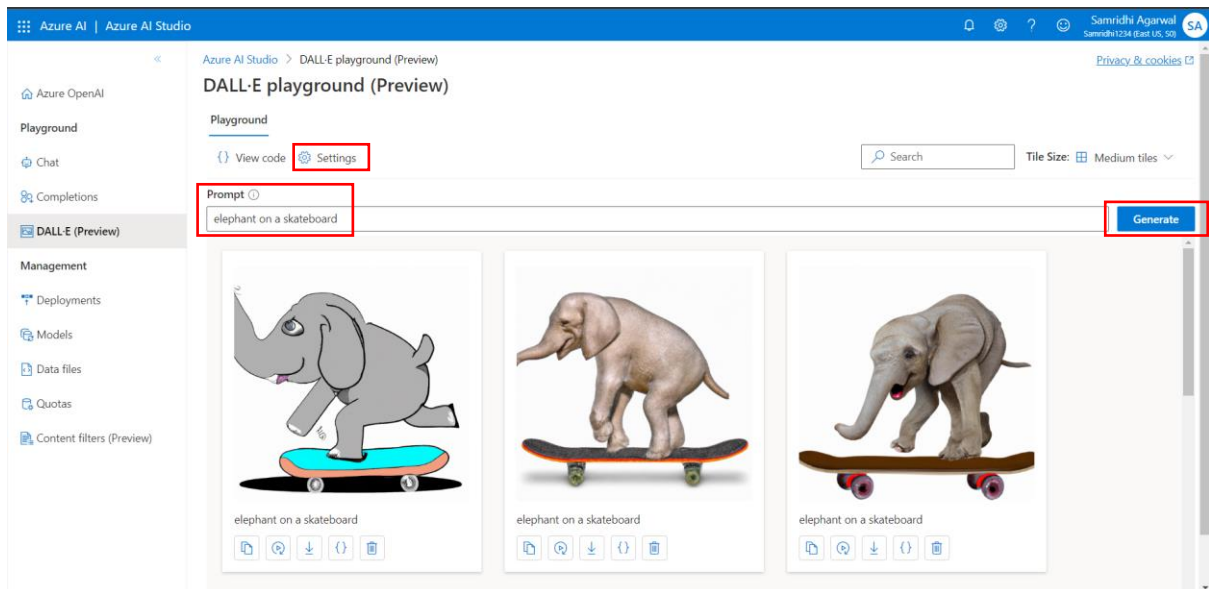
- To access Azure OpenAI Studio in a new browser, utilize the explore button on the overview page of the resource in the Azure portal. Go to Azure OpenAI Studio directly.



- Select DALL-E Playground preview in Azure OpenAI Studio.



- In the prompt box, enter the description of an image to generate. Then click generate and view the generated images.



Note: The number and size of image(s) can be set from the settings.

Using the Azure OpenAI REST API to consume DALL-E Model

- One can use the Azure OpenAI Service REST API to consume DALL-E model from applications.
- To make a REST call to the service, one needs the endpoint and authorization key for the Azure OpenAI Service resource one has provisioned in Azure. One initiates the image generation process by submitting a POST request to the service endpoint with the authorization key in the header. The request must contain the following parameters in a JSON body:

Prompt: The description of the image to be generated.

n: The number of images to be generated.

size: The resolution of the image(s) to be generated (256x256, 512x512, or 1024x1024)

For example, the following JSON could be used to generate an 512x512 image of badger wearing a tuxedo:

```
body.json > ...
1  {
2    "prompt": "A badger wearing a tuxedo",
3    "n": 1,
4    "size": "512x512"
5  }
6
```

The first request doesn't instantly yield picture generating results. Instead, the response contains an operation-location header with a callback service URL that application code may poll until image generating results are available.

When the operation has succeeded, a response like the following JSON is returned:

```
{ } result.json > ...
1  {
2    "created": 1686780744,
3    "expires": 1686867152,
4    "id": "6d765598-eeee-4f49--885d-03ee1c8f9b02",
5    "result": {
6      "created": 1686780744,
7      "data": [
8        {
9          "url": "https://dalleprouse..... .png"
10       }
11     ]
12   },
13   "status": "succeeded"
14 }
15
```

The result element contains URL elements that point to prompt-generated PNG image files.

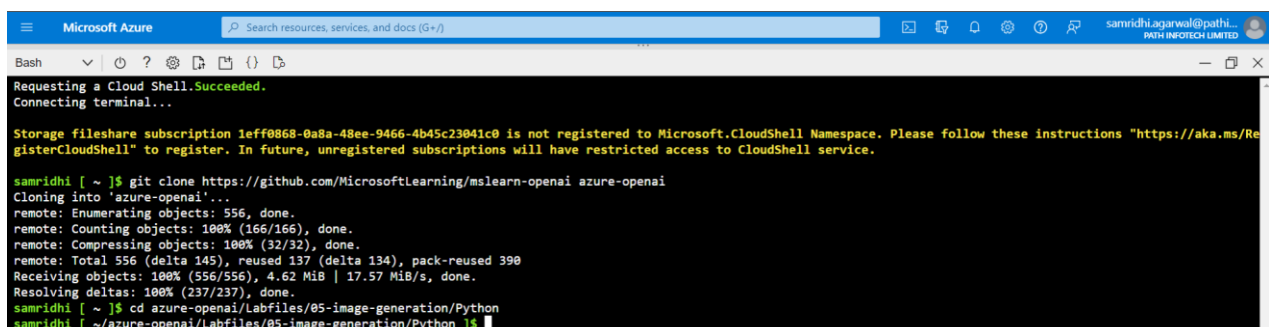
Image generation using REST API

The Azure OpenAI Services provides a REST API which can be used to submit prompts for content generation – including images generated by a DALL-E model.

Preparing the App Environment

In this, one would be using python to generate images by calling the REST API. Thus code is run in the cloud shell console interface in Azure portal.

01. In the cloud shell, we choose bash.
02. As the terminal starts, enter the following commands to download the application code to work with.
 - > git clone <https://github.com/MicrosoftLearning/mslearn-openai> azure-openaiThe files are downloaded to a folder named azure-openai.
03. Navigate to the folder for the language one wants to use by running the appropriate command.
 - > cd azure-openai/Labfiles/05-image-generation/Python



```
Microsoft Azure
Search resources, services, and docs (G+)
samridhi.agarwal@pathi...
PATH INFOTECH LIMITED

Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Storage fileshare subscription 1eff0862-0a8a-48ee-9466-4b45c23041c0 is not registered to Microsoft.CloudShell Namespace. Please follow these instructions "https://aka.ms/RegisterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service.

samridhi [ ~ ]$ git clone https://github.com/MicrosoftLearning/mslearn-openai azure-openai
Cloning into 'azure-openai'...
remote: Enumerating objects: 556, done.
remote: Counting objects: 100% (166/166), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 556 (delta 145), reused 137 (delta 134), pack-reused 390
Receiving objects: 100% (556/556), 4.62 MiB | 17.57 MiB/s, done.
Resolving deltas: 100% (237/237), done.
samridhi [ ~ ]$ cd azure-openai/Labfiles/05-image-generation/Python
samridhi [ ~/azure-openai/Labfiles/05-image-generation/Python ]$
```

04. Use the following command to open the built-in code editor and see the code files.

```
> pip install python-dotenv
```

Then, to update to the latest version.

```
> pip install --upgrade pip
```

```
samridhi [ ~/azure-openai/Labfiles/05-image-generation/Python ]$ pip install python-dotenv
Defaulting to user installation because normal site-packages is not writeable
Collecting python-dotenv
  Downloading python_dotenv-1.0.0-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
WARNING: The script dotenv is installed in '/home/samridhi/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
NOTE: The current PATH contains path(s) starting with '~', which may not be expanded by all applications.
Successfully installed python-dotenv-1.0.0

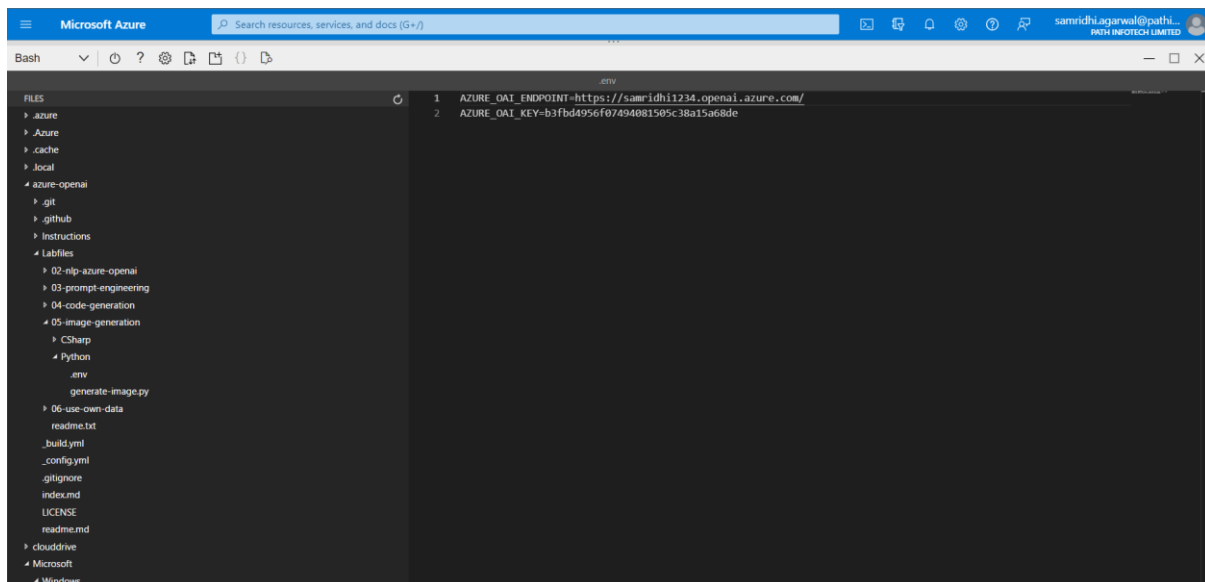
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
samridhi [ ~/azure-openai/Labfiles/05-image-generation/Python ]$ pip install --upgrade pip
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in /usr/lib/python3.9/site-packages (23.1.2)
Collecting pip
  Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
    2.1/2.1 MB 5.4 MB/s eta 0:00:00
Installing collected packages: pip
WARNING: The scripts pip, pip3 and pip3.9 are installed in '/home/samridhi/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
NOTE: The current PATH contains path(s) starting with '~', which may not be expanded by all applications.
Successfully installed pip-23.2.1

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
samridhi [ ~/azure-openai/Labfiles/05-image-generation/Python ]$
```

Note: In the console prompt pane, ensure the current folder is ~/azure-openai/Labfiles/05-image-generation/Python

Configuring the app

05. Save the endpoint and key in a file, named .env



The screenshot shows a code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with folders like 'azure', 'cache', 'local', 'azure-openai', 'Instructions', 'Labfiles', '02-nlp-azure-openai', '03-prompt-engineering', '04-code-generation', '05-image-generation', 'CSharp', 'Python', 'generate-image.py', '06-use-own-data', and 'cloudrive'. The code editor shows the content of the .env file:

```
.env
1 AZURE_OAI_ENDPOINT=https://samridhi1234.openai.azure.com/
2 AZURE_OAI_KEY=b3fbd4956f07494081505c38a15a68de
```

Viewing the app

06. While, reviewing the code file it must contain the following key features:

- The code makes https requests to the endpoint for your services, including the key for your service in the header. Both values are obtained from the configuration file.
- The code polls the callback URL until the status of the image-generation task is succeeded, and then extracts and displays a URL for the generated image.
- The process consists of 2 REST requests:
 - a) To initiate the image-generation request

b) To retrieve the results

The initial request includes the following data:

- i. The user-provided prompt describing the image to be generated.
- ii. The no. of images to be generated.
- iii. The resolution (size) of the image to be generated.
- iv. The response header from the initial request includes an operation-location value that is used for the subsequent callback to get the results.

```
generate-image.py
1 import requests
2 import time
3 import os
4 from dotenv import load_dotenv
5
6 def main():
7
8     try:
9         # Get Azure OpenAI Service settings
10        load_dotenv()
11        api_base = os.getenv("AZURE_OAI_ENDPOINT")
12        api_key = os.getenv("AZURE_OAI_KEY")
13        api_version = '2023-06-01-preview'
14
15        # Get prompt for image to be generated
16        os.system('clear')
17        prompt = input("Enter a prompt to request an image: ")
18
19        # Make the initial call to start the job
20        url = "{}openai/images/generations:submit?api-version={}".format(api_base, api_version)
21        headers= { "api-key": api_key, "Content-Type": "application/json" }
22        body = {
23            "prompt": prompt,
24            "n": 1,
25            "size": "512x512"
26        }
27        submission = requests.post(url, headers=headers, json=body)
28
29        # Get the operation-location URL for the callback
30        operation_location = submission.headers['Operation-Location']
31
32        # Poll the callback URL until the job has succeeded
33        status = ""
34        while (status != "succeeded"):
35            time.sleep(3)
36            response = requests.get(operation_location, headers=headers)
37            status = response.json()['status']
38
39        # Get the results
40        image_url = response.json()['result']['data'][0]['url']
41
42        # Display the URL for the generated image
43        print(image_url)
44
45    except Exception as ex:
46        print(ex)
47
48    if __name__ == '__main__':
49        main()
50
51
```


Running the app

07. In the console prompt area, write the following command

```
> python generate-image.py
```

08. When prompted, enter the description for the image.

```
Enter a prompt to request an image: animal party
https://dalleproduse.blob.core.windows.net/private/images/52db0e1e-d780-452e-a08f-9ace8668e797/generated_00.png?se=2023-08-25T06%3A27%3A58Z&sig=pEcK8A57v4QoB5IMxnd6MUzrAN3X
ECXb0uyUPdaQues%3D&sk=2023-08-30T18%3A52%3A52Z&skoid=09ba021e-c417-441c-b203-c81e5dcd7b7f&sk=s-b&skt=2023-08-23T18%3A52%3A52Z&sktid=33e01921-4d64-4f8c-a055-5bda9fd5e33d&skv
=2020-10-02&spr=https&sr=b&sv=2020-10-02
samridhi [ ~/azure-openai/Labfiles/05-image-generation/Python ]$
```

09. Wait for the image to be generated “-a hyperlink” will be displayed in the console pane. When selected, the hyperlink leads to a new web browser tab with the generated image.



Generated image

After doing thorough testing and implementing the necessary modifications, we have successfully launched the application. The system produces an appropriate generated picture based on the specified description.

SUMMARY

This module described the DALL-E image generation model and how one can use it in the Azure OpenAI Service to generate images based on natural language prompts. Thus, one can now explore DALL-E using the playground in Azure OpenAI Studio, and can use the REST API to build applications that uses DALL-E to generate new images.

REFERENCE

[Generate images with Azure OpenAI Service - Training | Microsoft Learn](#)

<https://learn.microsoft.com/en-us/training/modules/generate-images-azure-openai/>