gpt-35-turbo

Details Metrics Risks & Safety Consume

Language: Python | SDK: Azure OpenAI SDK | Authentication type: Entra ID Authentication

Get Started

Below are example code snippets for a few use cases. For additional information about Azure OpenAI SDK, see full documentation and samples

1. Authentication using Azure Active Directory Credentials

You can authenticate with Azure Active Directory using the Azure Identity library.

To use the DefaultAzureCredential provider shown below, or other credential providers provided with the Azure SDK, please install the azure-iden package:

If you're using bash:

```
pip install azure.identity
```

Authorization is easiest using DefaultAzureCredential. It finds the best credential to use in its running environment.

Set the values of the client ID, tenant ID, and client secret of the AAD application as environment variables: AZURE_CLIENT_ID, AZURE_TENANT_ID AZURE_CLIENT_SECRET.

```
export AZURE_CLIENT_ID="<AZURE_CLIENT_ID>"
export AZURE_TENANT_ID="<AZURE_TENANT_ID>"
export AZURE_CLIENT_SECRET="<AZURE_CLIENT_SECRET>"
```

2. Install dependencies

Install the Azure Open AI SDK using pip (Requires: Python >=3.8):

```
pip install openai
```

3. Run a basic code sample

This sample demonstrates a basic call to the chat completion API. The call is synchronous.

```
import os
from openai import AzureOpenAI
from azure.identity import DefaultAzureCredential, get_bearer_token_provider

endpoint = "https://rava-gpt-access.openai.azure.com/openai/deployments/gpt-35-turbo"
model_name = "gpt-35-turbo"
deployment = "gpt-35-turbo"
token_provider = get_bearer_token_provider(DefaultAzureCredential(), "https://cognitiveservices.azure.com/.
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    azure_ad_token_provider=token_provider,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
```

```
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment
)

print(response.choices[0].message.content)
```

4. Explore more samples

Run a multi-turn conversation

This sample demonstrates a multi-turn conversation with the chat completion API. When using the model for a chat application, you'll need to mana the history of that conversation and send the latest messages to the model.

```
import os
from openai import AzureOpenAI
from azure.identity import DefaultAzureCredential, get_bearer_token_provider

endpoint = "https://rava-gpt-access.openai.azure.com/openai/deployments/gpt-35-turbo"
model_name = "gpt-35-turbo"
deployment = "gpt-35-turbo"

token_provider = get_bearer_token_provider(DefaultAzureCredential(), "https://cognitiveservices.azure.com/.
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    azure_ad_token_provider=token_provider,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        },
        {
            "role": "assistant",
            "content": "Paris, the capital of France, is known for its stunning architecture, art museums, h
        },
        {
            "role": "user",
            "content": "What is so great about #1?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment
)

print(response.choices[0].message.content)
```

Stream the output

For a better user experience, you will want to stream the response of the model so that the first token shows up early and you avoid waiting for long responses.

```python
import os
from openai import AzureOpenAI
from azure.identity import DefaultAzureCredential, get_bearer_token_provider

endpoint = "https://rava-gpt-access.openai.azure.com/openai/deployments/gpt-35-turbo"
model_name = "gpt-35-turbo"
deployment = "gpt-35-turbo"

token_provider = get_bearer_token_provider(DefaultAzureCredential(), "https://cognitiveservices.azure.com/.
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    azure_ad_token_provider=token_provider,
)

response = client.chat.completions.create(
    stream=True,
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment,
)

for update in response:
    if update.choices:
        print(update.choices[0].delta.content or "", end="")

client.close()
```

5. Going beyond rate limits

The rate limits for the playground and free API usage are intended to help you experiment with models and prototype your AI application. For use be those limits, and to bring your application to scale, you must provision resources from an Azure account, and authenticate from there. You don't neec change anything else in your code.