Provision an Azure Resource for Speech Translation

To use the speech translation capabilities, you'll first need to provision a Speech resource in Azure.

Step 1: Create a Speech Resource

1. **Sign in to the Azure portal**: Azure Portal.
2. **Create a new resource**:
   - Click on the **"Create a resource"** button.
   - In the **"Search the Marketplace"** box, type **"Speech"** and select **"Speech"** from the list of results.
   - Click on **"Create"**.
3. **Configure your Speech resource**:
   - **Subscription**: Choose your Azure subscription.
   - **Resource Group**: Either create a new resource group or select an existing one.
   - **Region**: Choose the region closest to your users for lower latency.
   - **Name**: Provide a unique name for your Speech resource.
   - **Pricing Tier**: Select the appropriate pricing tier based on your usage requirements.
   - **Click "Review + create"**, then **"Create"**.
4. **Access your Speech resource**:

- After the resource is created, go to the resource page.
- **Copy the API key and endpoint**: You will need these to authenticate your API calls.

2. Using the Speech Translation API

Once your Speech resource is provisioned, you can start using the Speech Translation API to translate spoken language into text or speech in another language.

a. Real-Time Speech Translation with Python

You can use the Azure Cognitive Services Speech SDK to perform real-time speech translation. Here's a Python example:

Step 1: Install the Azure Speech SDK

Bash:

```
pip install azure-cognitiveservices-speech
```

Step 2: Create a Python Script for Translation

Python:

```
import azure.cognitiveservices.speech as speechsdk

# Set up your subscription info

speech_key = "Your-API-Key"

service_region = "Your-Region"


# Create a speech translation configuration

translation_config =
speechsdk.translation.SpeechTranslationConfig(subscription=speech_key,
region=service_region)


# Specify the source language and target languages

translation_config.speech_recognition_language = "en-US"

translation_config.add_target_language("es")  # Translate to Spanish

translation_config.add_target_language("fr")  # Translate to French


# Create a translation recognizer

audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)

translator =
speechsdk.translation.TranslationRecognizer(translation_config=translation_config,
audio_config=audio_config)


# Function to handle translation events

def translation_callback(evt):

    print(f"Recognized: {evt.result.text}")

    for language, translation in evt.result.translations.items():

        print(f"Translated into {language}: {translation}")


# Connect the event handler

translator.recognized.connect(translation_callback)
```

```
# Start continuous recognition
translator.start_continuous_recognition()

# Keep the recognition running
print("Listening... Press Ctrl+C to stop.")
try:
    while True:
        pass
except KeyboardInterrupt:
    translator.stop_continuous_recognition()
```

Explanation:

- **`speech_recognition_language`**: Sets the language of the speech input.
- **`add_target_language`**: Specifies the languages into which the speech should be translated.
- **`audio_config`**: Configures the audio input, in this case using the default microphone.

Running the Script:

- When you run the script, it listens for speech in the source language and provides translations in real time. The translations are printed to the console.

b. Translating Speech into Text

If you prefer translating speech into text, you can adapt the above code by removing the speech synthesis and focusing on the translated text outputs.

3. Speech Translation API with REST

You can also use the Speech Translation API via REST for scenarios where you might not be using the SDK. This involves making HTTP POST requests to the API endpoint with the appropriate headers and audio data.

Example REST API Call with `curl`

Here's a **`curl`** command that demonstrates how to perform speech translation:

Bash:

```
curl -X POST "https://<your-resource-
region>.api.cognitive.microsoft.com/sts/v1.0/issueToken" \

-H "Ocp-Apim-Subscription-Key: <your-api-key>" -d ""


token=$(curl -X POST "https://<your-resource-
region>.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?la
nguage=en-US&profanity=masked" \

-H "Authorization: Bearer $token" \

-H "Content-Type: audio/wav" \

--data-binary @path_to_audio_file.wav)


echo $token
```

Replace `<your-resource-region>` and **`<your-api-key>`** with your actual region and API key.

Replace `path_to_audio_file.wav` with the path to your audio file.

With the Azure AI Speech service, you can easily integrate real-time speech translation into your applications. This enables multilingual support, broadening your application's accessibility and user engagement across different languages. By provisioning the Speech resource and leveraging the API or SDK, you can start building sophisticated translation capabilities into your apps.

Translate speech to text

Before you can use the speech-to-text service, you need to provision an Azure resource for speech. This involves creating a Speech resource in the Azure portal, which provides you with the necessary API keys and endpoint URLs to access the service.

1. **Sign in to the Azure portal**: Azure Portal.
2. **Create a new resource:**
    - Click on "**Create a resource**".
    - Search for "**Speech**" in the marketplace and select "**Speech**".
    - Click on "**Create**".
3. **Configure your resource:**

- Subscription: Choose your subscription.
- Resource Group: Create or select an existing resource group.
- Region: Select the region closest to your users.
- Name: Provide a name for your Speech resource.
- Pricing Tier: Select the appropriate pricing tier.
- Click "Review + create", then "Create".

4. **Access the resource**:

- After the resource is created, go to the resource page and copy the **API key** and **endpoint URL**.

2. Using the Azure Speech-to-Text Service

Once you have provisioned your resource, you can use the Speech-to-Text API or SDK to convert spoken language into text.

a. Using the Speech SDK in Python

Here's how you can use the Azure Speech SDK to perform speech-to-text conversion in Python:

Step 1: Install the Azure Speech SDK

Bash:

```
pip install azure-cognitiveservices-speech
```

Step 2: Create a Python Script for Speech-to-Text

```
import azure.cognitiveservices.speech as speechsdk

# Set up your subscription info

speech_key = "Your-API-Key"

service_region = "Your-Region"



# Create a speech configuration

speech_config = speechsdk.SpeechConfig(subscription=speech_key,
region=service_region)



# Set up audio input from the default microphone

audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)



# Create a speech recognizer

speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
audio_config=audio_config)



# Function to handle recognized speech

def recognized_handler(evt):

    print(f"Recognized: {evt.result.text}")



# Connect the event handler

speech_recognizer.recognized.connect(recognized_handler)
```

```
# Start continuous recognition
speech_recognizer.start_continuous_recognition()

# Keep the recognition running
print("Listening... Press Ctrl+C to stop.")
try:
    while True:
        pass
except KeyboardInterrupt:
    speech_recognizer.stop_continuous_recognition()
```

Explanation:

- **`speechsdk.SpeechConfig`**: Configures the connection to the Azure Speech service using your subscription details.
- **`speechsdk.SpeechRecognizer`**: This object handles the speech recognition process.
- **`recognized_handler`**: This function is triggered every time speech is recognized, and it prints the recognized text.

Running the Script:

- When you run the script, it listens for speech through your microphone and outputs the recognized text in real-time.

b. Using the REST API

Alternatively, you can use the REST API to send audio data to the Azure Speech service and receive the transcribed text.

Example REST API Call with `curl`:

Bash:

```
curl -X POST "https://<your-region>.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?language=en-US" \

-H "Ocp-Apim-Subscription-Key: <your-api-key>" \

-H "Content-Type: audio/wav" \

--data-binary @path_to_audio_file.wav
```

Replace `<your-region>` with your Azure resource region.

Replace `<your-api-key>` with your Azure Speech API key.

Replace `path_to_audio_file.wav` with the path to your audio file.

This will return the transcribed text in the response.

3. Customizing the Speech-to-Text Process

The Azure Speech-to-Text service allows you to customize various aspects of the recognition process, including:

- **Language**: Specify the language of the speech.
- **Profanity Filter**: Mask or remove profane words.
- **Punctuation**: Automatically insert punctuation in the transcribed text.

By provisioning a Speech resource in Azure and using the Speech SDK or REST API, you can easily convert spoken language into text. This feature is highly useful for building voice-enabled applications, automating transcription tasks, and enhancing accessibility. With support for multiple languages and real-time recognition, Azure's Speech-to-Text service is a robust solution for modern applications.

Synthesize translations

Synthesize Translations refers to the process of converting translated text into spoken language using text-to-speech (TTS) technology. With Azure AI services, you can translate speech from one language to another and then synthesize the translated text into speech, enabling multilingual spoken communication in real-time.

1. Overview of the Process

The general process for synthesizing translations involves:

1. **Speech Recognition**: Convert spoken language (speech) to text in the source language.
2. **Text Translation**: Translate the recognized text into the desired target language.
3. **Speech Synthesis**: Convert the translated text into speech using TTS.

2. Prerequisites

To perform speech synthesis of translated text, you'll need:

- **Azure Speech resource**: Provisioned in the Azure portal.
- **Azure Translator resource**: (Optional) If you need to handle the translation step separately.
- **Azure SDK** or **REST API**: For implementing the synthesis in your application.

3. Step-by-Step Implementation

a. Provision an Azure Speech Resource

As previously mentioned, you need to create a Speech resource in the Azure portal to get the necessary API keys and endpoints.

b. Recognize and Translate Speech

You can start by recognizing speech in one language and translating it into another. This is typically done using the Azure Speech SDK or the Translator service.

Here’s an example in Python that integrates speech recognition, translation, and synthesis.

c. Python Example: Speech-to-Speech Translation and Synthesis

Python:

```
import azure.cognitiveservices.speech as speechsdk

# Set up your Azure subscription key and region

speech_key = "Your-Speech-API-Key"

service_region = "Your-Region"


# Configure the Speech service

translation_config =
speechsdk.translation.SpeechTranslationConfig(subscription=speech_key,
region=service_region)

translation_config.speech_recognition_language = "en-US"  # Source language

translation_config.add_target_language("es")  # Target language for translation (Spanish)


# Set up the audio input and translation recognizer

audio_config = speechsdk.audio.AudioConfig(use_default_microphone=True)

translator =
speechsdk.translation.TranslationRecognizer(translation_config=translation_config,
audio_config=audio_config)


# Function to handle the translation and synthesis

def translation_synthesis_callback(evt):

    print(f"Recognized: {evt.result.text}")

    for language, translation in evt.result.translations.items():

        print(f"Translated into {language}: {translation}")
        # Create a speech configuration for synthesis
        speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)
        speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config)

        # Synthesize the translated text
        synthesis_result = speech_synthesizer.speak_text_async(translation).get()

        # Check the synthesis result
        if synthesis_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:
            print(f"Synthesized speech for {language} completed successfully.")
        elif synthesis_result.reason == speechsdk.ResultReason.Canceled:
            print(f"Speech synthesis canceled: {synthesis_result.cancellation_details.reason}")
# Connect the event handler to the translation recognizer
translator.recognized.connect(translation_synthesis_callback)
# Start continuous recognition
translator.start_continuous_recognition()

# Keep the recognition running
print("Listening... Press Ctrl+C to stop.")
try:
    while True:
        pass
except KeyboardInterrupt:
    translator.stop_continuous_recognition()
```

4. Explanation of the Code

1. **Translation Configuration**:
   - **`speech_recognition_language`**: Sets the language of the input speech.
   - **`add_target_language("es")`**: Adds Spanish as the target language for translation.
2. **Translation Recognition**:
   - The `**TranslationRecognizer**` listens to the microphone input, recognizes speech, and translates it into the target language.
3. **Speech Synthesis**:
   - For each recognized and translated text, the `**SpeechSynthesizer**` converts the translated text into speech.
4. **Event Handling**:
   - The `**translation_synthesis_callback**` function handles recognized text, translation, and synthesis events, printing the results and indicating success or failure.

5. Advanced Customizations

- **Voice Selection**: You can customize the synthesized voice by specifying a different voice name in the **SpeechConfig**.
- **SSML (Speech Synthesis Markup Language)**: Use SSML to control aspects like pitch, rate, and volume of the synthesized speech.
- **Multiple Languages**: You can translate and synthesize in multiple languages by adding more target languages in the **add_target_language()** method.

6. Using the REST API for Speech Synthesis

If you prefer using the REST API, you can first translate the text using the Translator API and then send the translated text to the Text-to-Speech endpoint:

Bash:

```
curl -X POST "https://<your-region>.tts.speech.microsoft.com/cognitiveservices/v1" \

-H "Ocp-Apim-Subscription-Key: <your-api-key>" \

-H "Content-Type: application/ssml+xml" \

-H "X-Microsoft-OutputFormat: riff-24khz-16bit-mono-pcm" \

--data "<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='es-ES'><voice name='es-ES-AlvaroNeural'>Hola, este es un ejemplo de síntesis de texto traducido.</voice></speak>" \

--output output.wav
```

Synthesis of translations using Azure AI Speech services enables you to convert recognized and translated speech into spoken output in a target language. This can be particularly useful for building multilingual communication systems, voice assistants, or accessible applications that cater to users speaking different languages. By combining speech recognition, translation, and synthesis, you can create seamless speech-to-speech translation experiences.

Use below link for reference purpose

https://learn.microsoft.com/en-us/training/modules/translate-speech-speech-service/