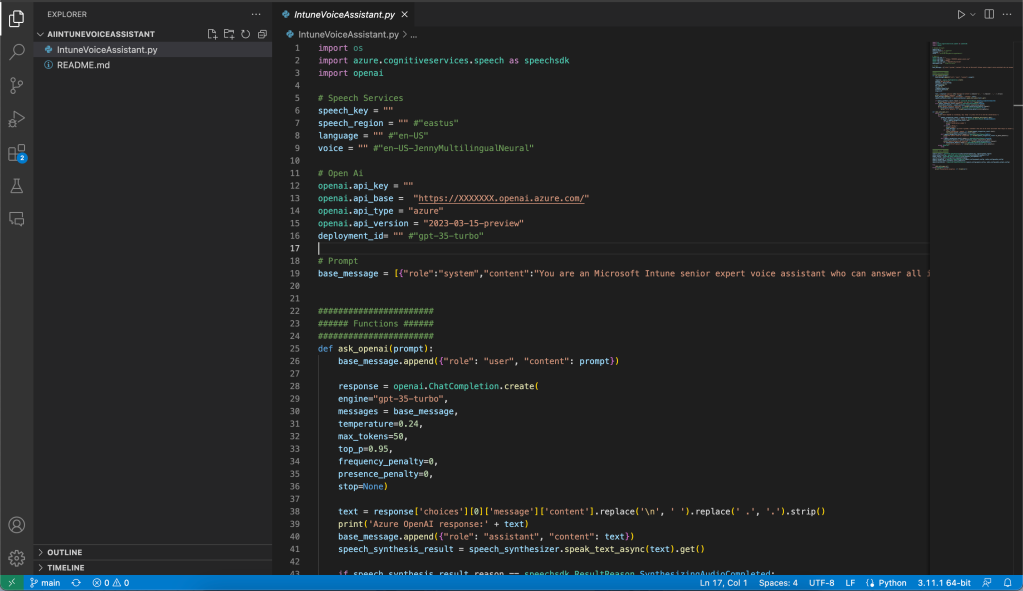
Intune AI Voice Bot

In the todays blog, I will announce the release of our experimental AI-driven voice assistant for Microsoft Intune-related questions. As you know, I am an huge fans of automation and AI technologies. I teamed up with [Fabian Peschke](https://github.com/Fadope1) to develop this innovative voice bot that aims to help users with their Intune questions.

Our voice bot is built using two different Microsoft cognitive services: *Azure Speech Services* and *OpenAI’s GPT-35 Turbo*. The Azure Speech Services allows the bot to recognize and synthesize speech, while OpenAI’s engine enables the bot to understand and respond to user queries intelligently. This bot was developed based on this [example from microsoft](https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/openai-speech?tabs=macos&pivots=programming-language-python).



Content

1. [Content](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#content)
2. [Requirements](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#requirements)
3. [How it works](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#how-it-works)
   1. [Azure Speech services](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#azure-speech-services)
   2. [OpenAI’s services](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#openai-s-services)
   3. [Integration of Azure Speech Services and GPT-35 Turbo Engine](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#integration-of-azure-speech-services-and-gpt-35-turbo-engine)
4. [Where can I find the script](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#where-can-i-find-the-script)
5. [How to setup](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#how-to-setup)
   1. [Set up Azure Speech Services](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#set-up-azure-speech-services)
   2. [Set up OpenAI’s GPT-35 Turbo Engine](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#set-up-openai-s-gpt-35-turbo-engine)
   3. [Configure the Script](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#configure-the-script)
6. [Conclusion](https://jannikreinhard.com/2023/04/23/intune-ai-voice-bot/#conclusion)

Requirements

* Python 3.6 or higher
* Azure Cognitive Services Speech SDK
* OpenAI Python library

You can install the required libraries using pip:

pip install azure-cognitiveservices-speech  
pip install openai

How it works

In this chapter, we will dive into how the AI-driven voice assistant works for Microsoft Intune. We will check the integration of Azure Speech Services and OpenAI’s engine and explain how they work together to provide a seamless user experience.

Azure Speech services

Azure Speech Services is a suite of APIs provided by Microsoft that facilitates speech recognition and synthesis. In our voice assistant, we use two main components of Azure Speech Services:

**Speech Recognition**: This component enables the voice assistant to transcribe spoken words into text. It listens to the user’s voice and converts the speech into text, which can then be processed by the OpenAi engine.

**Speech Synthesis**: This component is responsible for converting the text-based responses generated by the GPT-35 Turbo engine into spoken words. It uses a neural text-to-speech system to synthesise human-like speech, allowing the voice assistant to deliver answers audibly.

OpenAI’s services

The OpenAi engine is a powerful language model developed by OpenAI. It can understand and generate human-like text based on a given prompt. In our voice assistant, the GPT-35 Turbo engine processes the text generated by the speech recognition component and generates an appropriate response based on the user’s query.

Integration of Azure Speech Services and GPT-35 Turbo Engine

The script we’ve developed integrates Azure Speech Services and OpenAI’s GPT-35 Turbo engine. Here’s a step-by-step explanation of how the process works:

* The user initiates the conversation with the voice assistant by saying “**Hey**” followed by their question.
* The speech recognition component of Azure Speech Services transcribes the user’s speech into text.
* The text is passed to the GPT-35 Turbo model, which processes the input and generates an appropriate response based on the briefing message and the base message defined in the script.

*You are an Microsoft Intune senior expert voice assistant who can answer all intune related questions. You are friendly and concise. You only provide factual answers to queries, and do not provide answers that are not related to Microsoft products or intune.*

* The generated response is then passed to the speech synthesis component, which converts the text into speech and delivers the answer audibly.
* If the user wishes to end the conversation, they can say “**Stop**” or press Ctrl-Z. To reset the conversation and delete the history, the user can say “**Reset**.”

Where can I find the script

As always you can find my script in my [github repositroy](https://github.com/JayRHa/AiIntuneVoiceAssistant) or here:

import os

import azure.cognitiveservices.speech as speechsdk

import openai

# Speech Services

speech\_key = ""

speech\_region = "" #"eastus"

language = "" #"en-US"

voice = "" #"en-US-JennyMultilingualNeural"

# Open Ai

openai.api\_key = ""

openai.api\_base = "https://XXXXXXX.openai.azure.com/"

openai.api\_type = "azure"

openai.api\_version = "2023-03-15-preview"

deployment\_id= "" #"gpt-35-turbo"

# Prompt

base\_message = [{"role":"system","content":"You are an Microsoft Intune senior expert voice assistant who can answer all intune related questions. You are friendly and concise. You only provide factual answers to queries, and do not provide answers that are not related to Microsoft products or intune."}]

#######################

###### Functions ######

#######################

def ask\_openai(prompt):

base\_message.append({"role": "user", "content": prompt})

response = openai.ChatCompletion.create(

engine="gpt-35-turbo",

messages = base\_message,

temperature=0.24,

max\_tokens=50,

top\_p=0.95,

frequency\_penalty=0,

presence\_penalty=0,

stop=None)

text = response['choices'][0]['message']['content'].replace('\n', ' ').replace(' .', '.').strip()

print('Azure OpenAI response:' + text)

base\_message.append({"role": "assistant", "content": text})

speech\_synthesis\_result = speech\_synthesizer.speak\_text\_async(text).get()

if speech\_synthesis\_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:

print("Speech synthesized to speaker for text [{}]".format(text))

elif speech\_synthesis\_result.reason == speechsdk.ResultReason.Canceled:

cancellation\_details = speech\_synthesis\_result.cancellation\_details

print("Speech synthesis canceled: {}".format(cancellation\_details.reason))

if cancellation\_details.reason == speechsdk.CancellationReason.Error:

print("Error details: {}".format(cancellation\_details.error\_details))

def chat\_with\_open\_ai():

while True:

print("Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.")

try:

speech\_recognition\_result = speech\_recognizer.recognize\_once\_async().get()

if speech\_recognition\_result.reason == speechsdk.ResultReason.RecognizedSpeech:

text = speech\_recognition\_result.text

if text == "Stop.":

print("Conversation ended.")

break

if text == "Reset.":

print("Reset")

base\_message = [{"role":"system","content":"You are an AI voice assistant that helps to answer questions."}]

if "Hey" in text:

print("Recognized speech: {}".format(speech\_recognition\_result.text))

ask\_openai(speech\_recognition\_result.text)

elif speech\_recognition\_result.reason == speechsdk.ResultReason.NoMatch:

print("No speech could be recognized: {}".format(speech\_recognition\_result.no\_match\_details))

break

elif speech\_recognition\_result.reason == speechsdk.ResultReason.Canceled:

cancellation\_details = speech\_recognition\_result.cancellation\_details

print("Speech Recognition canceled: {}".format(cancellation\_details.reason))

if cancellation\_details.reason == speechsdk.CancellationReason.Error:

print("Error details: {}".format(cancellation\_details.error\_details))

except EOFError:

break

#######################

######## Start ########

#######################

speech\_config = speechsdk.SpeechConfig(subscription=speech\_key, region=speech\_region)

audio\_output\_config = speechsdk.audio.AudioOutputConfig(use\_default\_speaker=True)

audio\_config = speechsdk.audio.AudioConfig(use\_default\_microphone=True)

speech\_config.speech\_recognition\_language=language

speech\_recognizer = speechsdk.SpeechRecognizer(speech\_config=speech\_config, audio\_config=audio\_config)

speech\_config.speech\_synthesis\_voice\_name=voice

speech\_synthesizer = speechsdk.SpeechSynthesizer(speech\_config=speech\_config, audio\_config=audio\_output\_config)

try:

chat\_with\_open\_ai()

except Exception as err:

print("Encountered exception. {}".format(err))

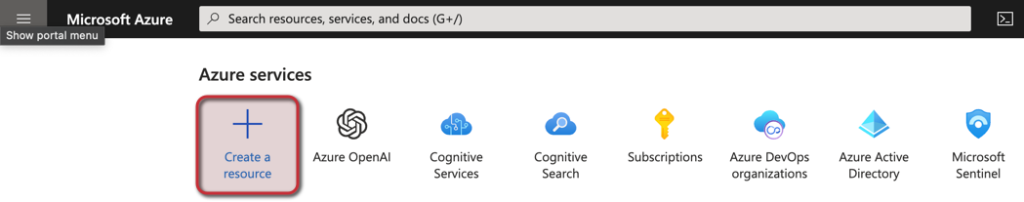
How to setup

As descripted in the chapter “how it works” we need two cognitive services. In this chapter I will show you how you can add this both services in azure.

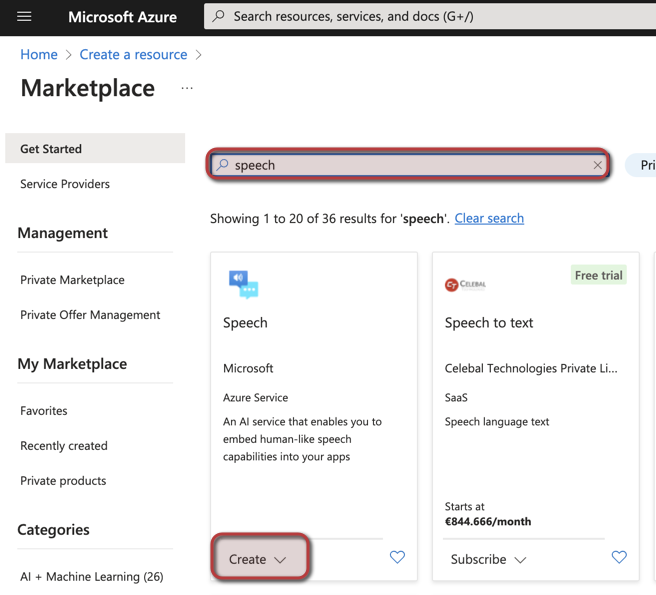
Set up Azure Speech Services

To set up Azure Speech Services, follow these steps:

* Sign in to the Azure portal (<https://portal.azure.com/>)
* Click *Create a resource*

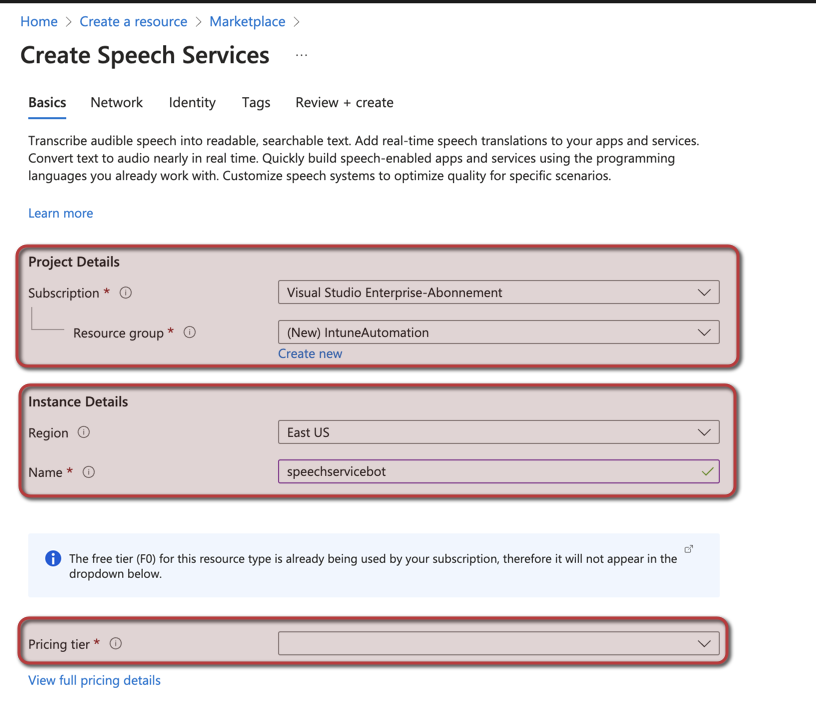


* In the search bar, type “Speech” and select “Speech” from the search results.
* Click on the “Create” button to start the setup process.

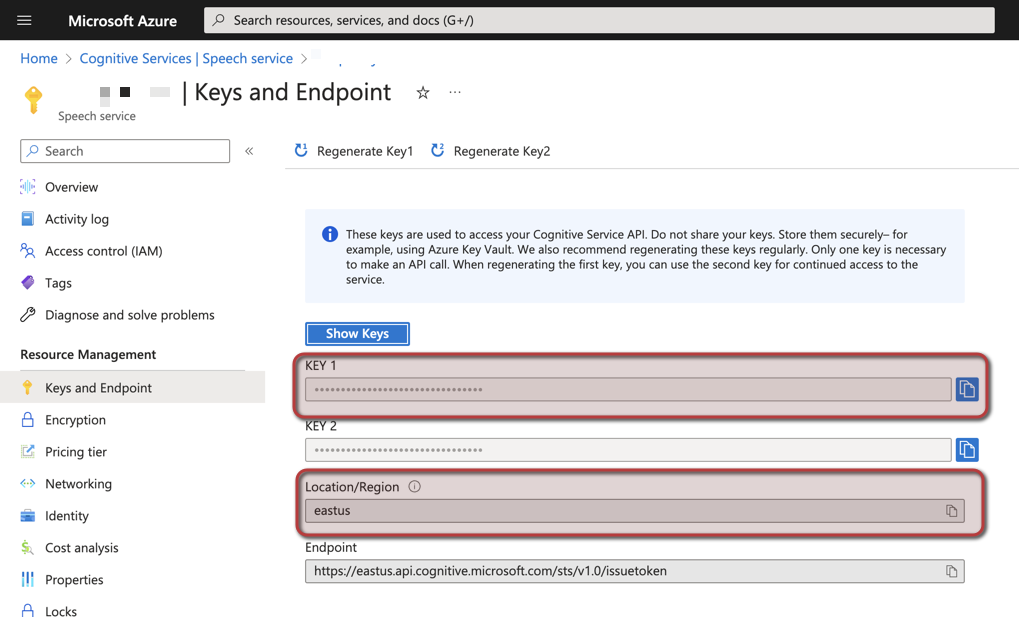


* Fill in the required fields, including the **subscription**, **resource group**, **name**, **region**, and **pricing tier**. Then click “***Review*** ***+ create***“

If this is your first resource you can select the price tier F0 to have some free contingent



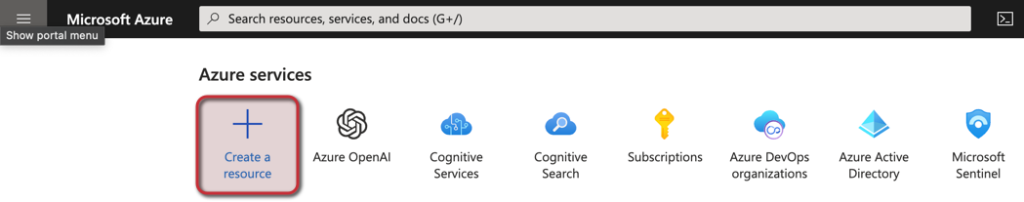
* After reviewing your settings, click “Create” to deploy the Speech service.
* Once the deployment is complete, navigate to the “Keys and Endpoint” section of the Speech resource. Make a note of the “**Key1**” and the “**Location/Region**” as you will need them later to configure the script.



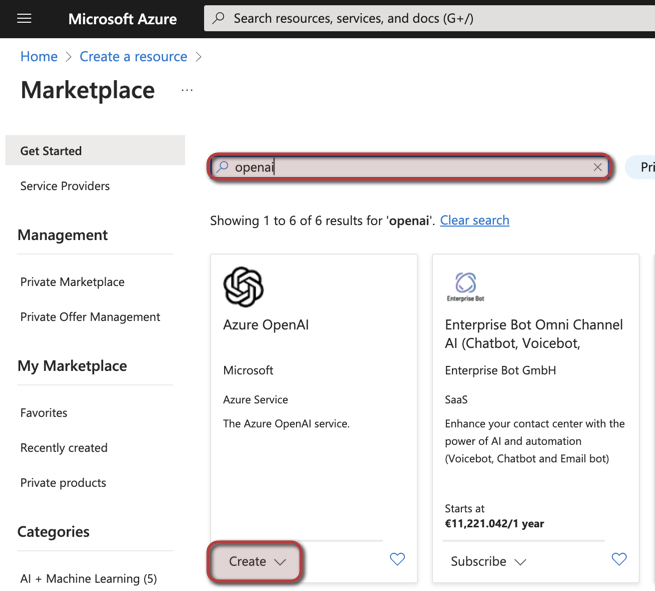
Set up OpenAI’s GPT-35 Turbo Engine

To set up the GPT-35 Turbo engine, you will need access to OpenAI’s API. Follow these steps:

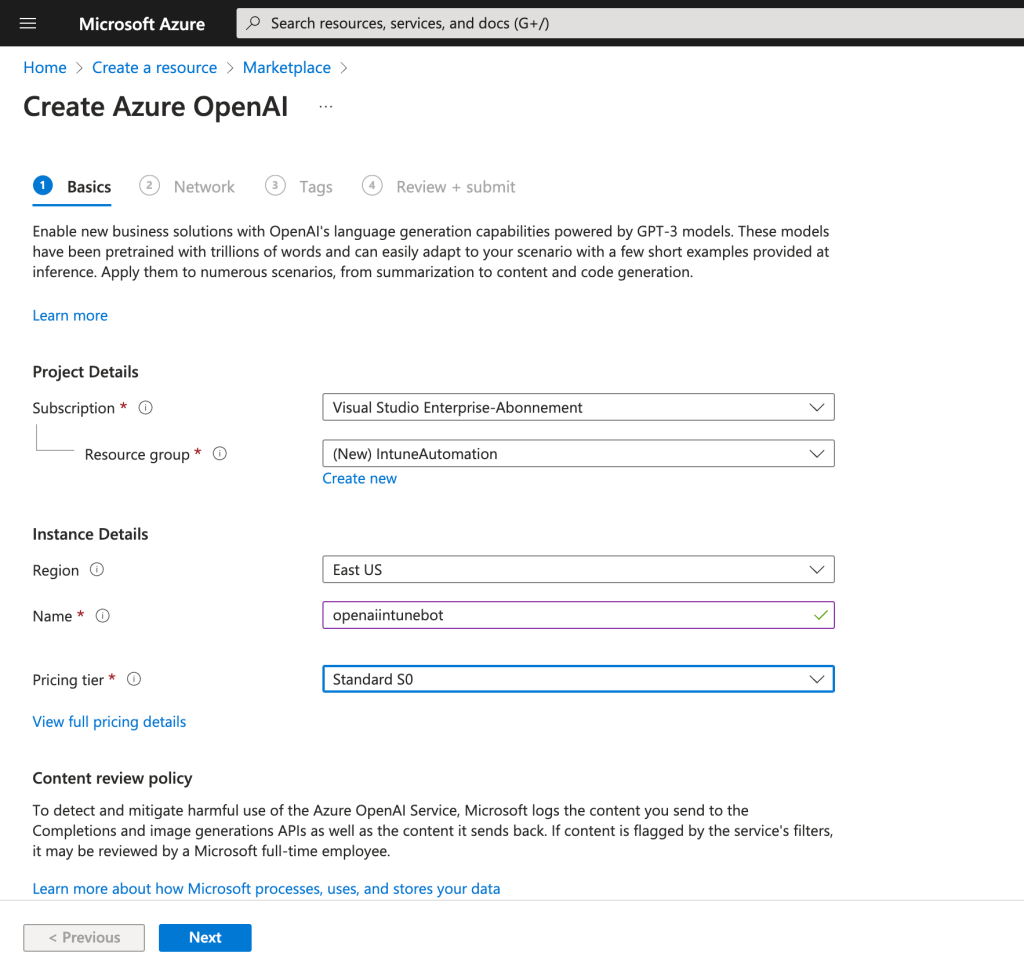
* Fill out the following [formula](https://customervoice.microsoft.com/Pages/ResponsePage.aspx?id=v4j5cvGGr0GRqy180BHbR7en2Ais5pxKtso_Pz4b1_xUOFA5Qk1UWDRBMjg0WFhPMkIzTzhKQ1dWNyQlQCN0PWcu) to get access to the open ai beta. (It could take some days until this is approved)
* Sign in to the Azure portal (<https://portal.azure.com/>)
* Click *Create a resource*



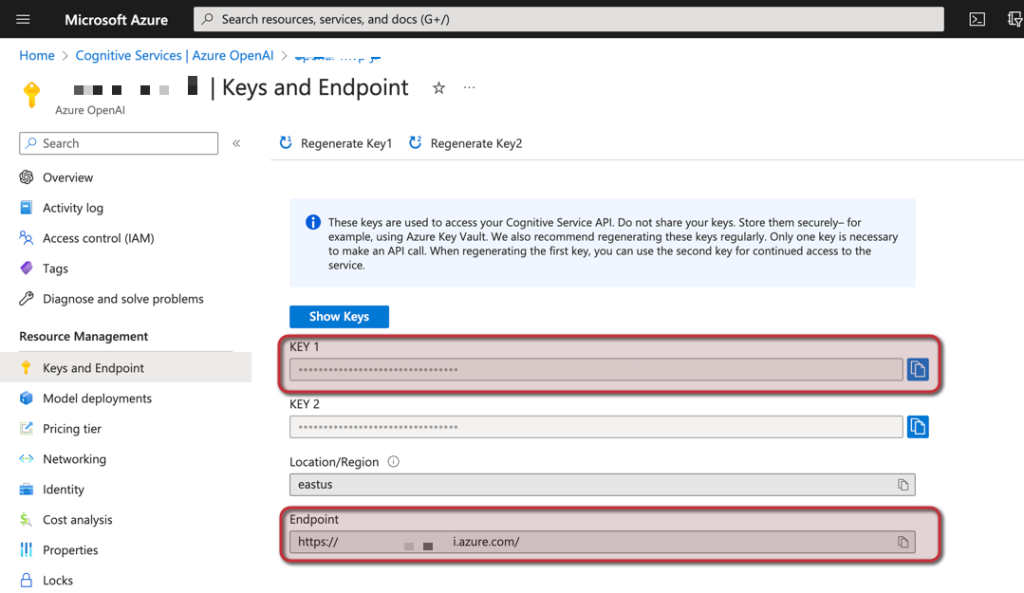
* In the search bar, type “OpenAi” and select “OpenAi” from the search results.
* Click on the “Create” button to start the setup process.



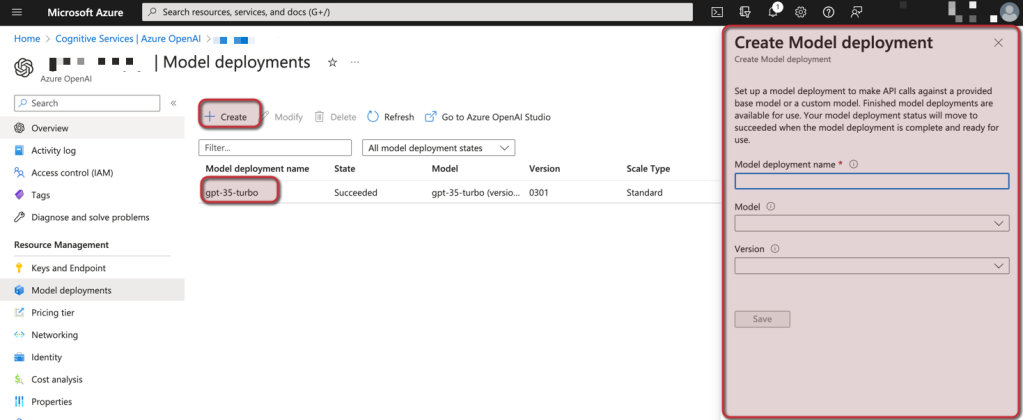
* Fill in the required fields, including the **subscription**, **resource group**, **name**, **region**, and **pricing tier**. Then click “***Review*** ***+ create***“



* After reviewing your settings, click “**Create**” to deploy the OpenAi service.
* Once the deployment is complete, navigate to the “Keys and Endpoint” section of the Speech resource. Make a note of the “**Key1**” and the “**Endpoint**” as you will need them later to configure the script.



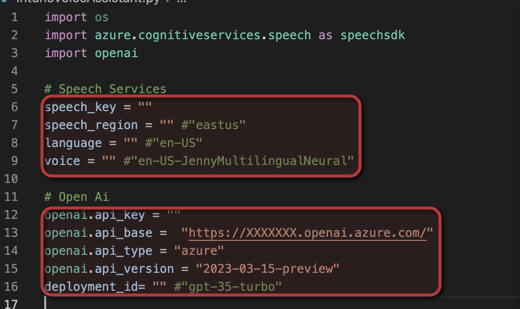
* Click on “**Model deployment**” and **+Create**
* Deploy the **gpt-35-turbo** model



Configure the Script

Now that you have both Azure Speech Services and the OpenAI’s services set up, you need to configure the script with the required API keys and endpoints.

* Open the script in your preferred code editor e.g. vs code.
* Fill in the noted values in the variables



* **speech\_key**: Your Azure Cognitive Services Speech subscription key.
* **speech\_region**: The region of your Azure Cognitive Services Speech service (e.g., “eastus”).
* **language**: The language for speech recognition and synthesis (e.g., “en-US”).
* **voice**: The voice for speech synthesis (e.g., “en-US-JennyMultilingualNeural”).
* **openai.api\_key**: Your OpenAI API key.
* **openai.api\_base**: The base URL for the OpenAI API (e.g., “[https://XXXXXXX.openai.azure.com/&#8221](https://xxxxxxx.openai.azure.com/&#8221);).
* **deployment\_id**: The deployment ID for GPT-3.5 Turbo (e.g., “gpt-35-turbo”).

You can find more informations here: <https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support?tabs=stt>

* Save the changes and run the Script

You can run the script on you pc or you can build you own voice assistant with help of an raspberry pi, an speeker and an microphone.

Conclusion

With the script configured, you can now run the voice assistant. Follow the instructions in the script to start a conversation with the bot and ask your Microsoft Intune-related questions.

You have also a lot of possibilities to further develop this or to simplify you daily work or to answer your questions.

That’s it! You have successfully set up both cognitive services and configured the script for the AI-driven voice assistant. Enjoy your new Intune voice assistant and let me know if you have any questions or feedback.

Analyze the document completely with references and give me bullet pointed steps , on how to implement the same for a large organization, give me accurate details with each bullet point that you are writing. Note: I want to write detailed article on this topic, how this can help a origination. Give me detailed and accurate description on this topic.

ChatGPT

**Step-by-Step Implementation of Intune AI Voice Assistant for a Large Organization**

1. **Understand the Purpose of the Intune AI Voice Bot**:
   * The Intune AI Voice Bot serves as a voice-based assistant for Microsoft Intune queries, automating responses to user questions about device and configuration management. Using both Azure Speech Services and OpenAI’s GPT-3.5 Turbo, it recognizes voice commands and generates relevant responses, creating an interactive and hands-free way to manage Intune tasks​(Intune AI Voice).
2. **Prerequisites Setup**:
   * **Python Environment**: Install Python 3.6 or higher on the system.
   * **Required Libraries**: Install necessary libraries using the following commands:

bash

Copy code

pip install azure-cognitiveservices-speech

pip install openai

* + **Azure Subscription**: Ensure access to Azure Cognitive Services and OpenAI API, which will be essential for setting up speech recognition and text generation components​(Intune AI Voice).

1. **Set Up Azure Speech Services**:
   * **Create Resource**: Sign in to the Azure portal, go to **Create a Resource**, and search for “Speech” under Azure Cognitive Services.
   * **Configure Resource Details**: Select **Subscription**, **Resource Group**, **Region**, and **Pricing Tier** (F0 for limited free usage).
   * **Obtain API Key and Endpoint**: After deployment, navigate to the “Keys and Endpoint” section, and record the “Key1” and “Location/Region” values. These will be needed to configure the voice assistant’s speech recognition and synthesis​(Intune AI Voice).
2. **Set Up OpenAI’s GPT-3.5 Turbo Engine**:
   * **Access OpenAI in Azure**: In the Azure portal, create an “OpenAI” resource by selecting the appropriate subscription, resource group, and region.
   * **Deploy the GPT-3.5 Turbo Model**: After creating the OpenAI resource, deploy the GPT-3.5 Turbo model in the **Model Deployment** section.
   * **Obtain API Key and Endpoint**: Once the deployment completes, go to **Keys and Endpoint**. Record the API key and base endpoint URL, which will be necessary for configuring the script​(Intune AI Voice).
3. **Configure the Voice Assistant Script**:
   * **Edit the Script**: Open the script in a code editor (e.g., VS Code). Input the following values:
     + speech\_key: Azure Cognitive Services Speech API key.
     + speech\_region: Azure Speech region (e.g., “eastus”).
     + language: Language code for speech recognition and synthesis (e.g., “en-US”).
     + voice: Specific voice for speech synthesis (e.g., “en-US-JennyMultilingualNeural”).
     + openai.api\_key: OpenAI API key.
     + openai.api\_base: Base URL for the OpenAI service (e.g., “https://XXXX.openai.azure.com/”).
     + deployment\_id: Deployment ID for GPT-3.5 Turbo (e.g., “gpt-35-turbo”).
   * **Save and Test**: Save changes, and test the script to ensure the voice assistant accurately processes queries and provides answers​(Intune AI Voice).
4. **Testing the Integration**:
   * **Run the Script**: Use a command like python your\_script.py to launch the voice assistant.
   * **Test Commands**: Interact with the assistant by initiating commands with “Hey” (e.g., “Hey, can you check the compliance status of devices?”).
   * **Real-Time Interaction**: Observe the assistant’s ability to convert speech to text, send queries to OpenAI, and synthesize responses back to spoken language​(Intune AI Voice).
5. **Deploy on a Dedicated Device (Optional)**:
   * For a hands-free experience, deploy the assistant on a device like a Raspberry Pi with an external microphone and speaker. This setup can be useful for larger organizations looking for centralized Intune support stations where users can access device information by simply speaking commands.
6. **Data Privacy and Security Measures**:
   * **Role-Based Access Control**: Ensure only authorized personnel can access and configure the bot by integrating Azure Active Directory roles.
   * **Compliance with Data Policies**: Confirm that data processed by the bot aligns with company policies and Microsoft’s data privacy standards, ensuring any speech or text data is securely managed.
7. **Benefits for Large Organizations**:
   * **Improved Productivity**: With a voice-activated assistant, IT staff can perform Intune-related tasks faster without manual input, freeing them up for more strategic initiatives.
   * **Streamlined Device Management**: The assistant can quickly respond to device compliance, application status, and configuration queries, helping IT teams monitor the organization’s device ecosystem efficiently.
   * **Enhanced User Experience**: By reducing wait times for device-related questions, the assistant improves response time for employee support requests, contributing to overall user satisfaction.
8. **Future Potential**:
   * As technology evolves, the assistant can incorporate more advanced natural language processing and even machine learning to handle increasingly complex tasks or integrate with other Microsoft services, further expanding its value.

**Detailed Benefits of Implementing the Intune AI Voice Assistant**

* **Automation of Routine Tasks**: By automating frequently asked questions, the assistant can handle common requests (e.g., checking device compliance or status updates), minimizing repetitive tasks and reducing the manual workload on IT teams.
* **Enhanced Support Efficiency**: In large organizations where immediate support is vital, the voice assistant provides real-time answers to employees and administrators, reducing response times and keeping operational efficiency high.
* **Scalable Solution**: The Azure-based voice bot is highly scalable, making it suitable for large organizations with numerous employees and devices, as it adapts to demand without needing extensive hardware.
* **Security and Compliance**: With Azure’s secure environment and strict access controls, the organization can deploy the assistant without risking sensitive data exposure, ensuring compliance with industry standards.

**Conclusion**

Implementing the Intune AI Voice Assistant in a large organization enables hands-free interaction with Intune, simplifies IT operations, and offers secure, scalable support. By integrating this assistant, organizations can empower their IT teams, improve response times, and enhance productivity across the board.