GUVI

GUVI

16, OCT - 2024

CLOUD COMPUTING - MICROSOFT AZURE

ZEN CLASS - MAIN PROJECT - 1

Name - Punitha Ramesh Keswani
Reg. Mail in GUVI - Punitha.tech@gmail.com
Batch - CC2WE-E

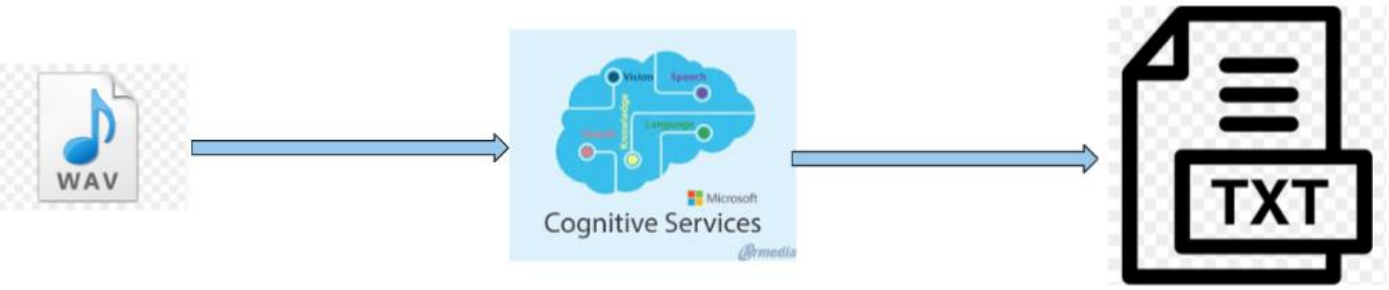
Project Scope

Create a demo using any of the cognitive services and showcase the use of that service:
Created Speech service in Azure Portal
Implementation Steps:

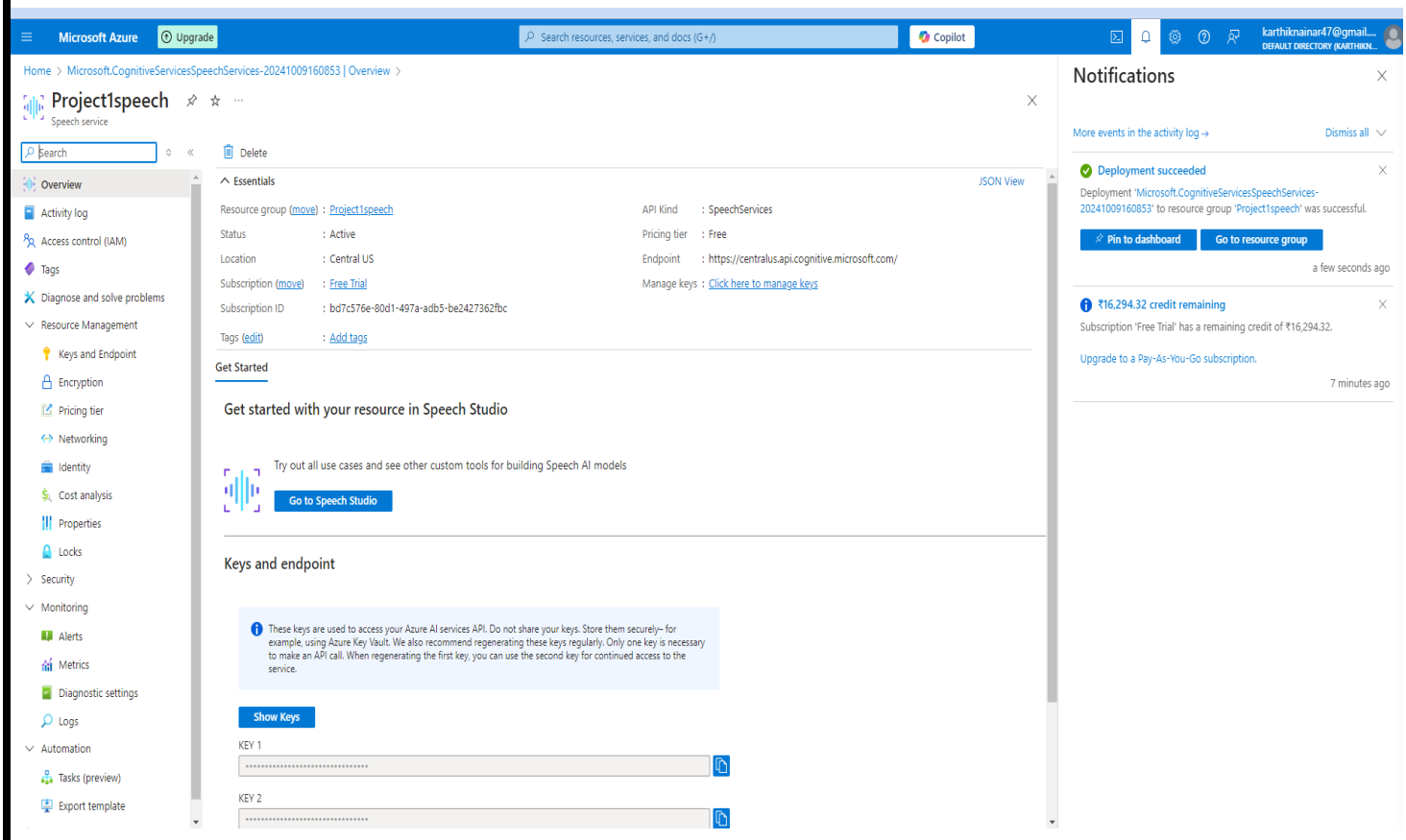
1. Create Azure Account: Sign up for an Azure account if you do not have one.
2. Deploy Linux VM: In the Azure portal, create a new Linux virtual machine (e.g., Ubuntu)
3. SSH into VM using putty
4. Install Dependencies:
   - Update the package list: sudo apt update && sudo apt upgrade.
   - Install Python and pip: sudo apt install python3-pip.
   - Install Azure Speech SDK: Use pip to install the SDK: pip3 install azure-cognitive services- speech.
5. Create Speech Resource: In Azure, create a Speech resource and copy the API key and endpoint.
6. Write Python Script: Create a new file (speech_to_text.py) and write code to interact with the Speech service.
7. Upload Audio File: Use scp to transfer audio files from your local machine to the VM.
8. Run the Script: Execute the Python script: python3 speech_to_text.py.

This streamlined process will help you set up and use Azure Speech-to-Text on a Linux VM efficiently.

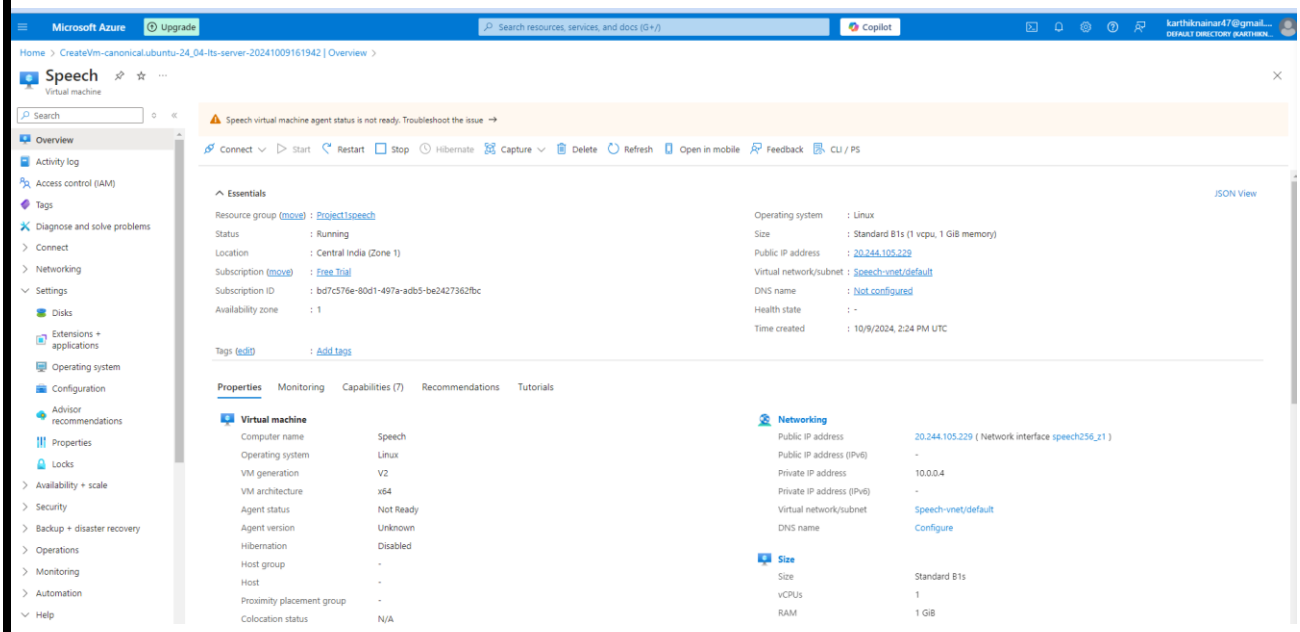
Project Architecture



- Login to Azure Portal
- **Create Speech service** in Central US
- Note your Endpoint and Key



- **Create Linux VM**

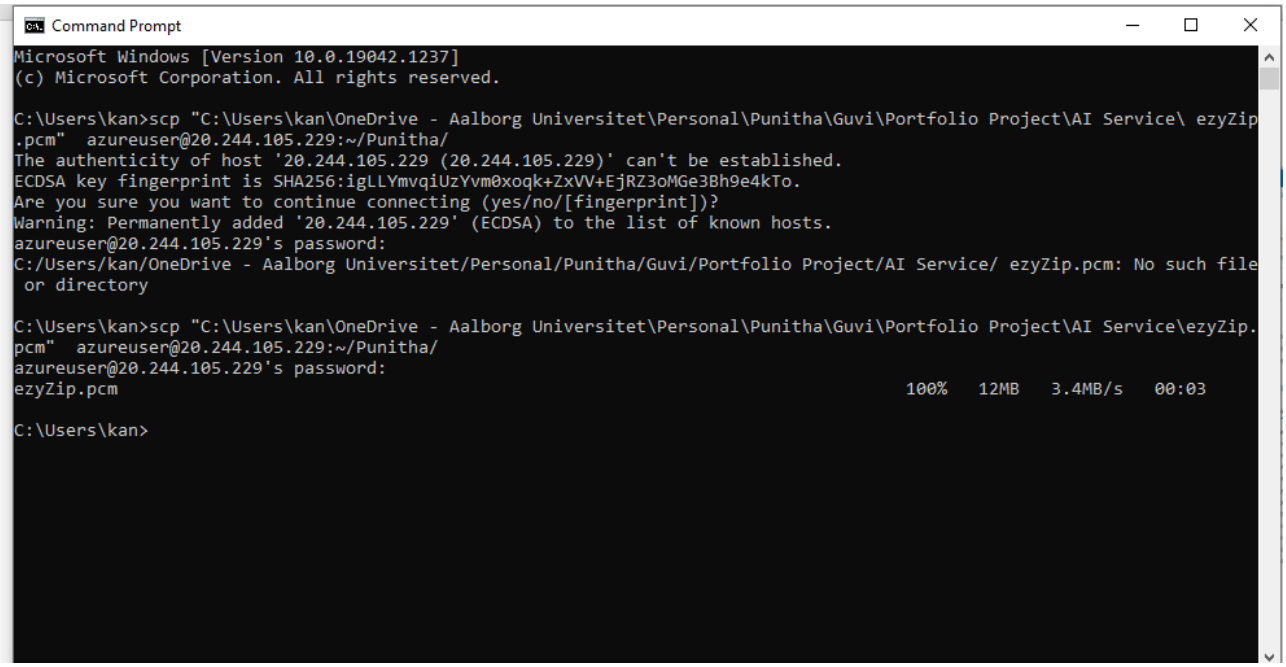


- Ssh linux box through putty
- Create a directory to save your file : Mkdir Punitha
- Cd Punitha
- Copy pcm format wav file from local to remote server

scp “C:\Users\kan\OneDrive - Aalborg Universitet\Personal\Punitha\Guvi\Portfolio Project\AI Service\Titanic.wav” azureuser@20.244.105.229:~/Punitha/

- Optional we can use python script to change .mp3 to .wav by using moviepy library

- .wav file loaded to remote server



- Use below commands to check audio is readable or not(Just in Case)

➢ Validate audio file has data or not
  - sudo apt install sox
  - sox Titanic.wav -n stat

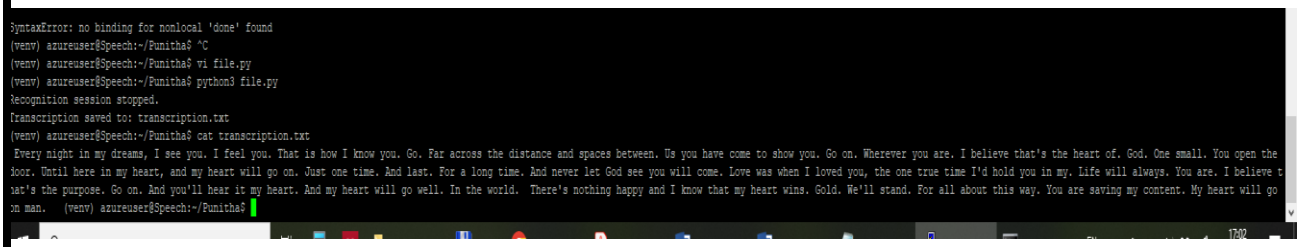
- **Create environment in your remote server to run python file by following below steps**
    - Sudo apt update
    - sudo apt install python3-pip
    - sudo apt-get install python3-venv
    - python3 -m venv venv
    - source venv/bin/activate
    - pip install azure-cognitiveservices-speech

- **Create python code which transfers your .wav to text file**
    - Create file transcribe.py by using vi transcribe.py
    - It includes below code

```
import azure.cognitiveservices.speech as speechsdk
import time
import os

# Set up the Azure Speech configuration
speech_key = "your key"
service_region = "centralus"
speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)

# Set the audio file path
audio_file = "Titanic.wav"

# Check if the audio file exists
if not os.path.isfile(audio_file):
    print(f"Error: The audio file '{audio_file}' does not exist.")
    exit(1)

# Set up the audio configuration
audio_config = speechsdk.audio.AudioConfig(filename=audio_file)

# Create a speech recognizer object
speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config, audio_config=audio_config)

# Create an empty list to store the transcription results
transcriptions = []

# Define an event handler for continuous recognition
def continuous_recognition_handler(evt):
    if evt.result.reason == speechsdk.ResultReason.RecognizedSpeech:
        transcriptions.append(evt.result.text)
    elif evt.result.reason == speechsdk.ResultReason.NoMatch:
        print("No speech could be recognized.")
    elif evt.result.reason == speechsdk.ResultReason.Canceled:
        print(f"Recognition canceled: {evt.result.cancellation_details.reason}")
```

```
# Connect the event handlers
speech_recognizer.recognized.connect(continuous_recognition_handler)

# Start continuous recognition
speech_recognizer.start_continuous_recognition()

# Wait for recognition to complete
done = False  # Declare 'done' as a global variable

def stop_recognition(evt):
    global done  # Use 'global' to modify the global variable
    print("Recognition session stopped.")
    done = True

speech_recognizer.session_stopped.connect(stop_recognition)

# Start the recognition and wait until session stops
while not done:
    time.sleep(0.5)  # This keeps the program alive until the recognition session completes

# Stop continuous recognition
speech_recognizer.stop_continuous_recognition()

# Combine transcriptions into a single string
transcription = ' '.join(transcriptions)

# Write the transcription to a file
output_file = "transcription.txt"
with open(output_file, "w") as file:
    file.write(transcription)

print("Transcription saved to:", output_file)
```

- Execute
  - transcribe.py



Here we ca n see the txt file and its content of our .wav file.