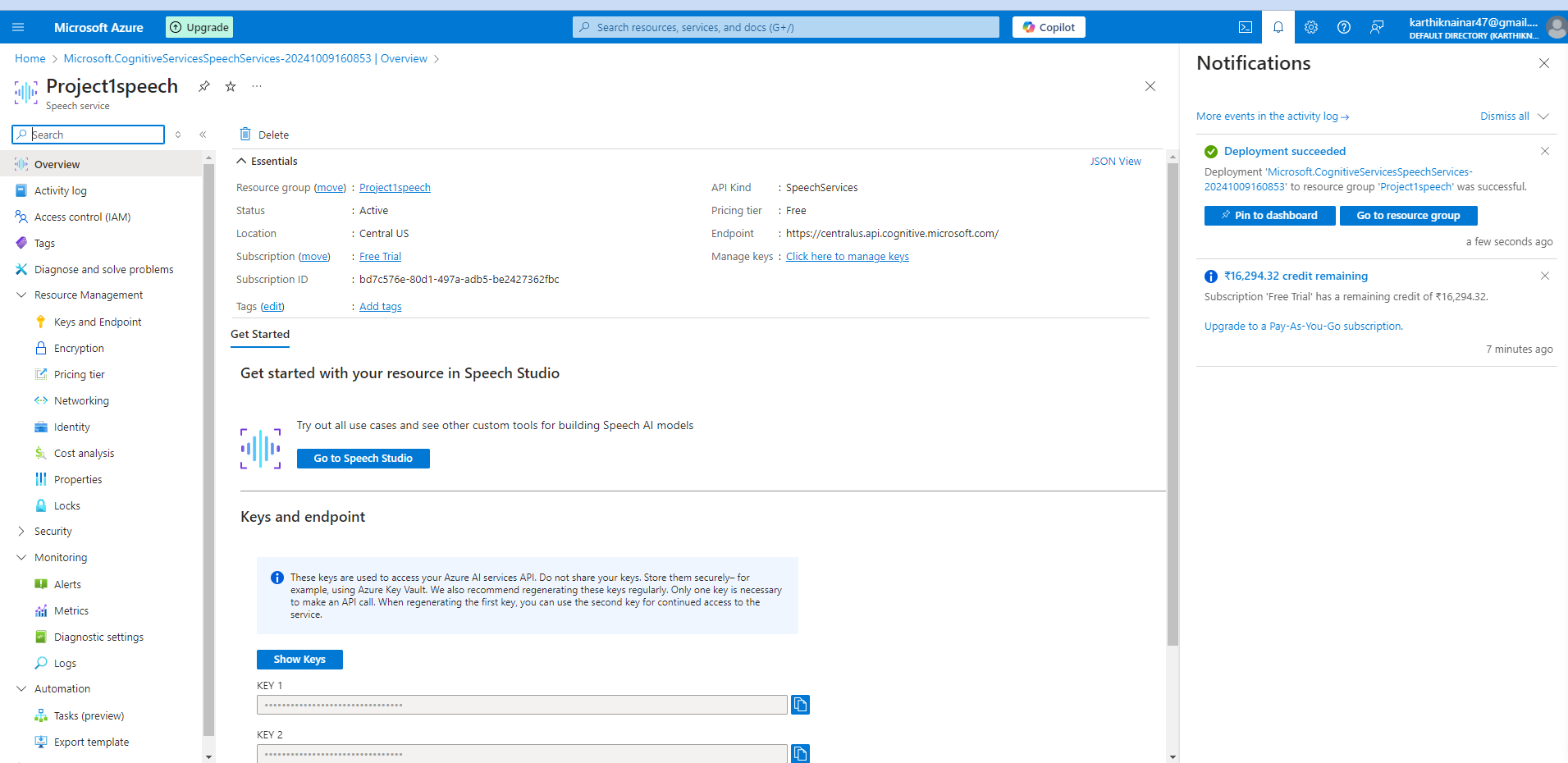
Project 1 : Create a demo using any of the cognitive services and showcase the use of that service. This is similar to what we did in class with compute vision.

Step 1 : Created Speech service in Azure Portal

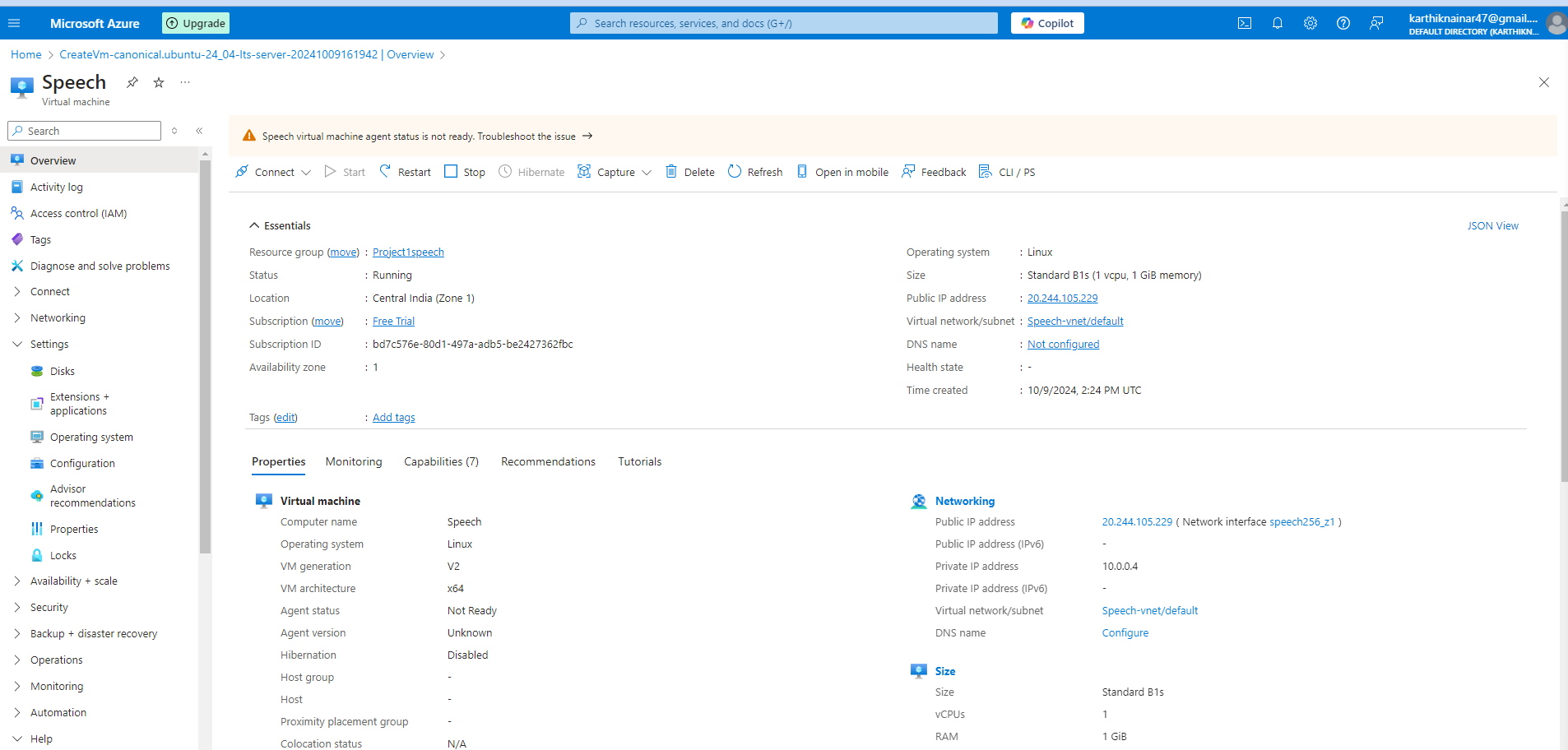


Endpoint and Key:

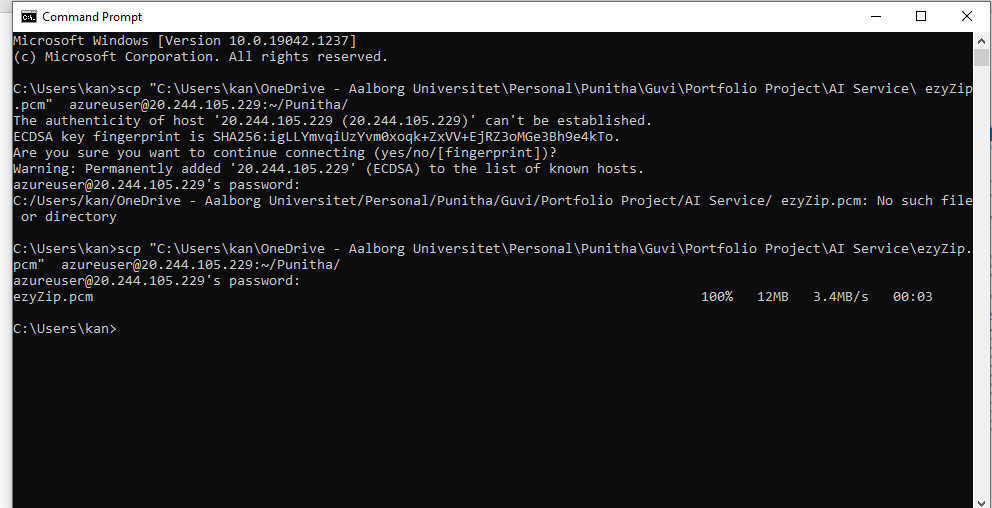
<https://centralus.api.cognitive.microsoft.com/>

93dccb9b4b664194a6db186686591827

Step 2 : Created Linux VM



* Ssh linux box through putty
* 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/](mailto: azureuser@20.244.105.229:~/Punitha/)



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

------------------------------------------------------------------------------------

To check audio file has data or not

sudo apt install sox

sox Titanic.wav -n stat

--------------------------------------------------------------------------------

vi file.py

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

Python3 file.py

