Assignment Objectives

Create a RESTful service. This service allows its users to upload audio files and specify the language of audio (it could be English or Hindi). The service extracts the following pieces of information from the audio file:

1. How many speakers were present in the audio. There can be up to two speakers.
2. What word was said at what time and by whom within the audio.
3. Dominant sentiment in the audio.

Since the audio processing can take a while, the API works asynchronously. When the user submits an audio file, the API returns a job-id. The user can query that job-id to find the status of audio processing. If the audio processing is still in progress, the query returns a response indicating that the file is still being processed. However, if the audio processing is over, the query returns the information extracted from the audio file.

Finally, write a document that describes your approach for solving the problem, what open source tools and cloud services you used, what audio files did you use for testing the code and the instructions for someone else to run your code.

Extra credit for packaging the service in a docker image.

Approach:

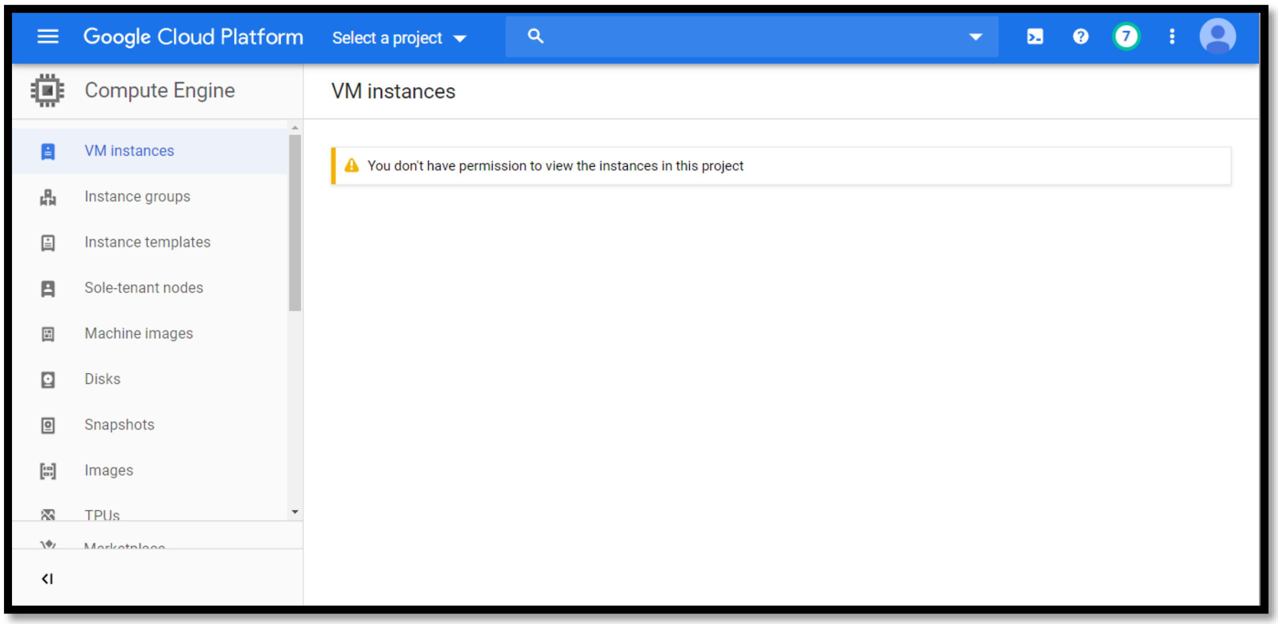
To achieve the project objectives followed the below approach:

1) Used Google Speech APIs for Speech-to text conversion and Speaker Diarization (Distinguishing Speakers in an audio file)
2) GCP Platform for development purposes – GCP Virtual Machines (Debian 10), Google Cloud Storage (for intermediate storage of data for asynchronous API requests) and Google Secret Manager for storing application access details.
3) Used Flask python based framework for APIs development.
4) RQ python library that uses Redis for queueing jobs and process them in background with workers
5) Postman to initiate requests and test REST API calls responses.
6) Created 2 simple audio files in .wav format. One each in English and Hindi languages
7) Packaged as Docker Image (work in progress)

Instructions to run the code:

1) Install postman. This will be used to fire REST API requests.

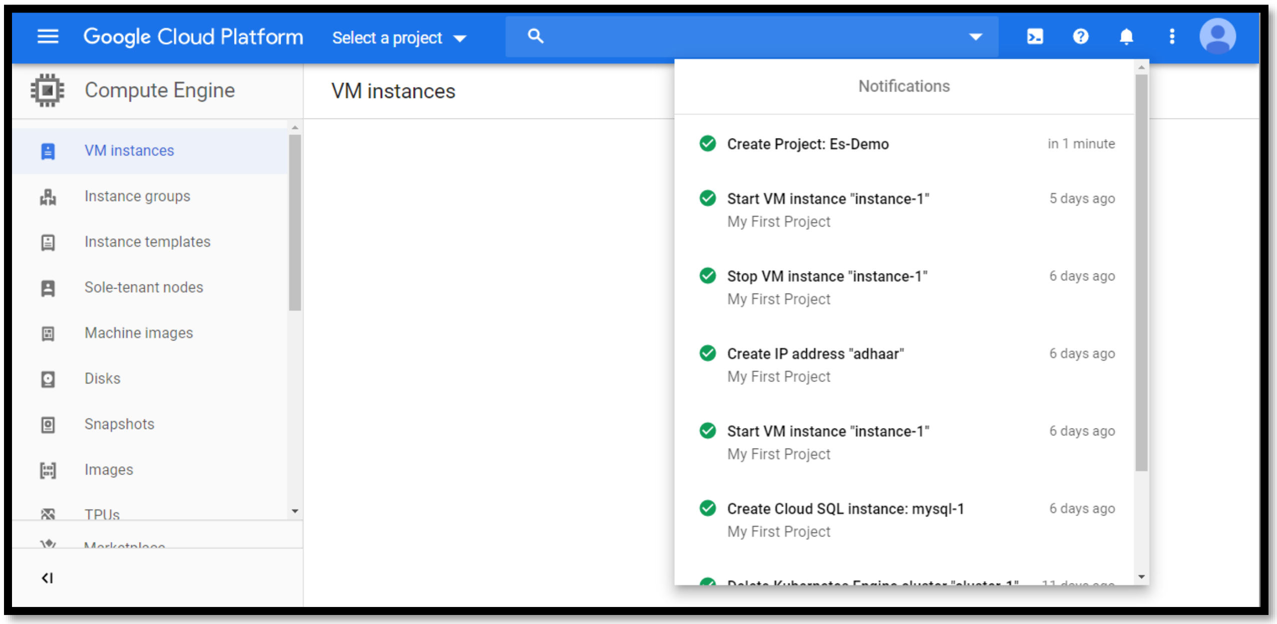
2) Login to your google cloud account and create a new project.



VM instances

You don't have permission to view the instances in this project

3) Once your project is created, select your project



VM instances

Notifications

Create Project: Es-Demo in 1 minute

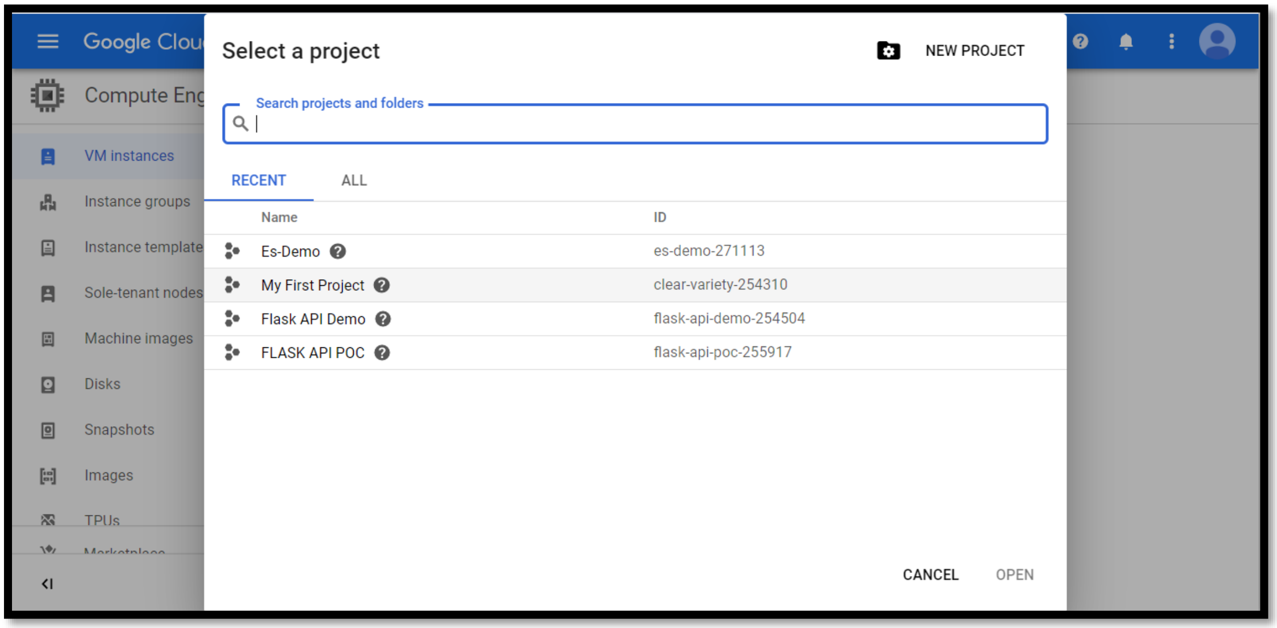
Start VM instance "instance-1" 5 days ago
My First Project

Stop VM instance "instance-1" 6 days ago
My First Project

Create IP address "adhaar" 6 days ago
My First Project

Start VM instance "instance-1" 6 days ago
My First Project

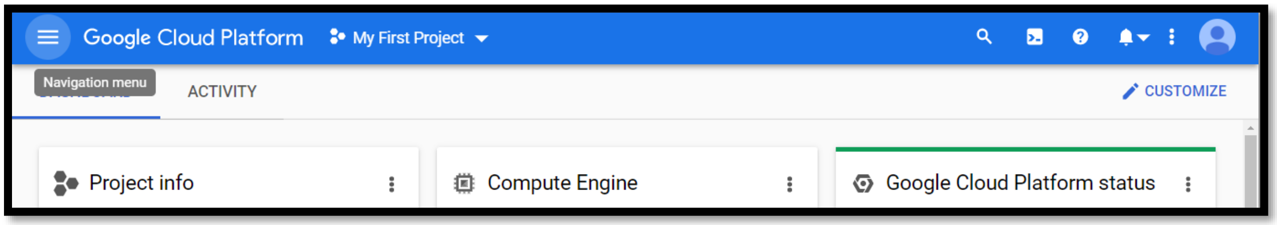
Create Cloud SQL instance: mysql-1 6 days ago
My First Project



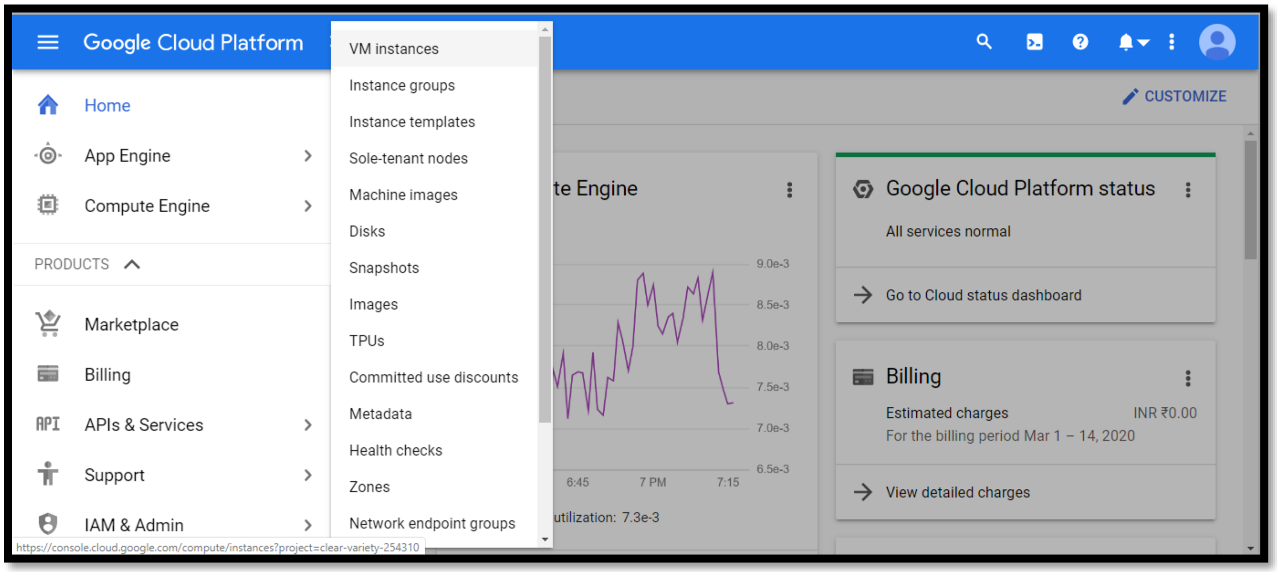
Name	ID
Es-Demo	es-demo-271113
My First Project	clear-variety-254310
Flask API Demo	flask-api-demo-254504
FLASK API POC	flask-api-poc-255917

4) Now we require a VM, which can host our application. Here we are going to use compute engine for our application. To create VM follow below mentioned steps:

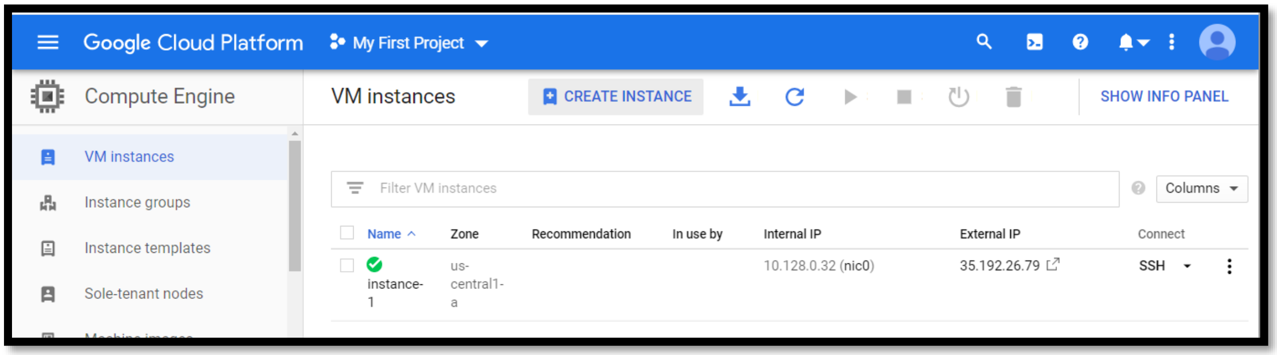
First go to Navigation menu



Hover over compute engine and select VM instances

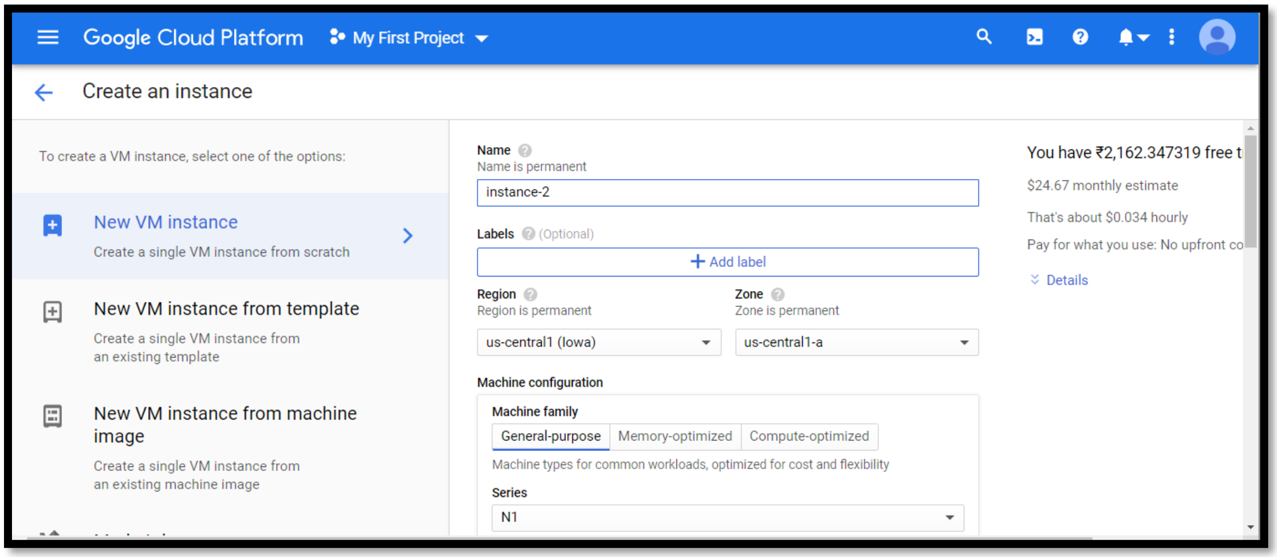


Now select **CREATE INSTANCE**



Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
instance-1	us-central1-a			10.128.0.32 (nic0)	35.192.26.79	SSH

Provide VM Name as per your need.


Google Cloud Platform
My First Project
Create an instance
To create a VM instance, select one of the options:
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
New VM instance from machine image
Create a single VM instance from an existing machine image
Add label
Details

Name
Name is permanent
instance-2

Labels (Optional)

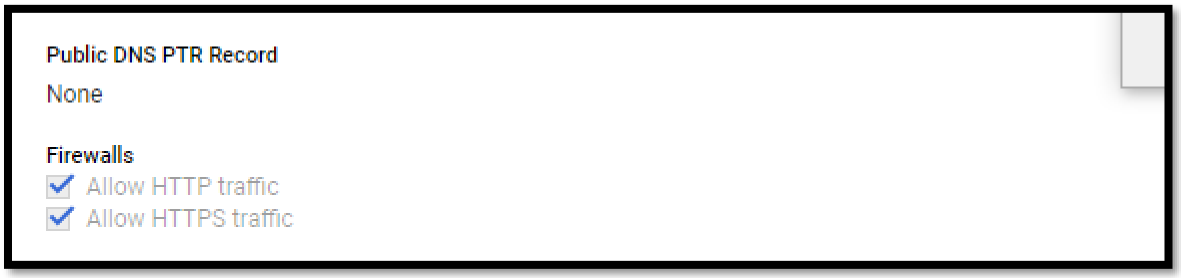
Region
Region is permanent
us-central1 (Iowa)

Zone
Zone is permanent
us-central1-a

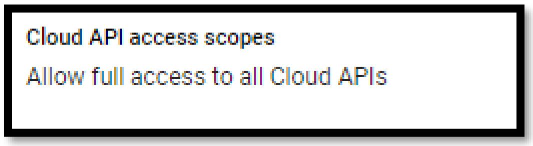
Machine configuration
Machine family
General-purpose | Memory-optimized | Compute-optimized
Machine types for common workloads, optimized for cost and flexibility
Series
N1

You have ₹2,162.347319 free t
$24.67 monthly estimate
That's about $0.034 hourly
Pay for what you use: No upfront co

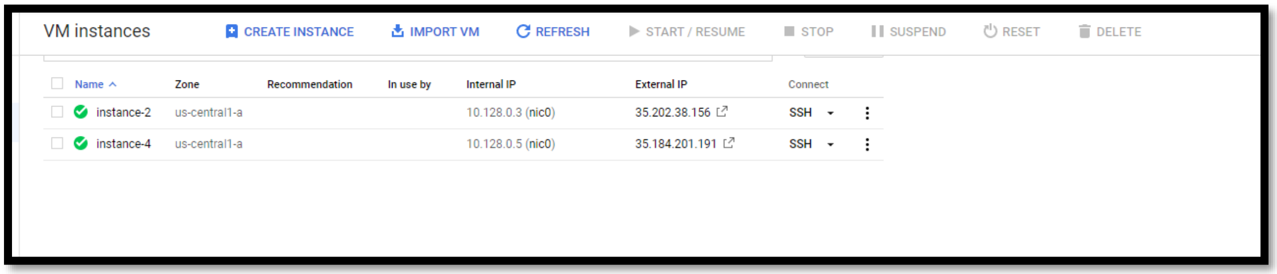
Please make sure to check below while creating VM instance in GCP.


Public DNS PTR Record
None

Firewalls
- [x] Allow HTTP traffic
- [x] Allow HTTPS traffic

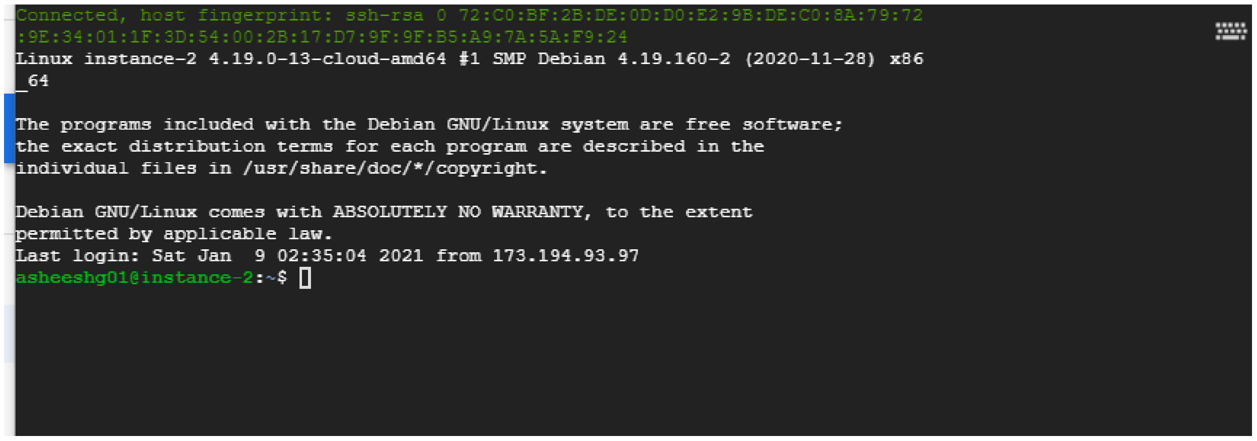

Cloud API access scopes
Allow full access to all Cloud APIs

You can see your instance once created



Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
instance-2	us-central1-a			10.128.0.3 (nic0)	35.202.38.156	SSH
instance-4	us-central1-a			10.128.0.5 (nic0)	35.184.201.191	SSH

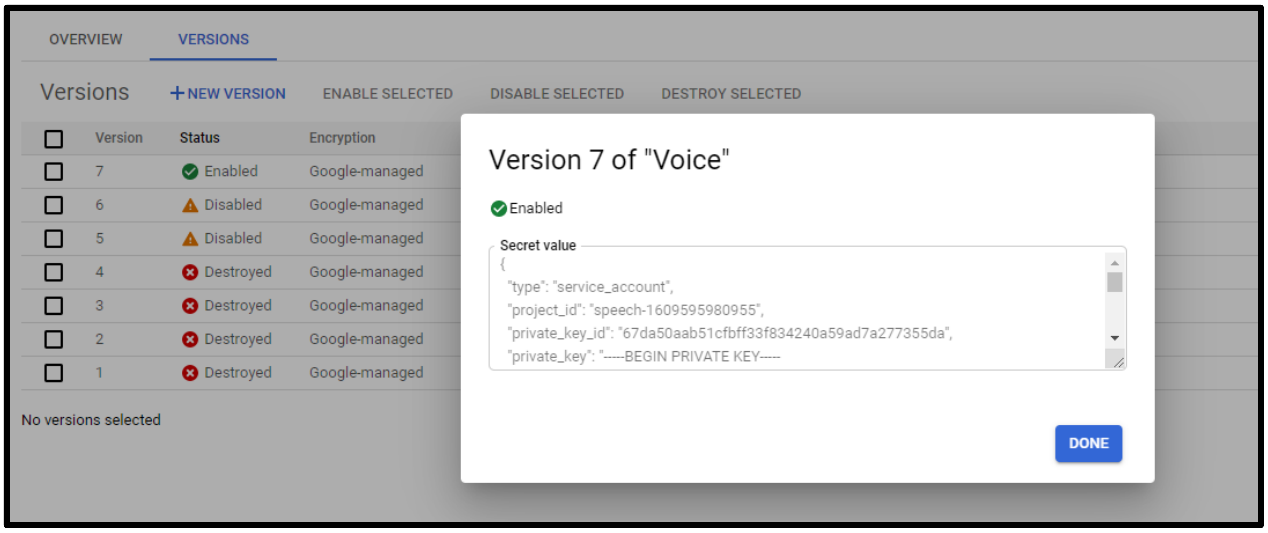
Click on SSH to open a terminal


```
Connected, host fingerprint: ssh-rsa 0 72:C0:BF:2B:DE:0D:D0:E2:9B:DE:C0:8A:79:72
:9E:34:01:1F:3D:54:00:2B:17:D7:9F:9F:B5:A9:7A:5A:F9:24
Linux instance-2 4.19.0-13-cloud-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86
_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan  9 02:35:04 2021 from 173.194.93.97
asheeshg01@instance-2:~$ 
```

5) Please load Speech-67da50aab51c.json in GCP secret Manager. Please make sure to tailor the parameters like bucketname etc. based on your GCP storage bucket where you want to store your output files.



Version	Status	Encryption
7	Enabled	Google-managed
6	Disabled	Google-managed
5	Disabled	Google-managed
4	Destroyed	Google-managed
3	Destroyed	Google-managed
2	Destroyed	Google-managed
1	Destroyed	Google-managed

Version 7 of "Voice"

Enabled

Secret value

```
{
  "type": "service_account",
  "project_id": "speech-1609595980955",
  "private_key_id": "67da50aab51cfbff33f834240a59ad7a277355da",
  "private_key": "-----BEGIN PRIVATE KEY-----
```

6) Install below libraries in VM
   a. Flask
   b. Flask-RESTful
   c. pydub
   d. wave
   e. config
   f. rq
   g. google-cloud-language
   h. google-cloud-secret-manager
   i. google-cloud-storage
   j. google-cloud-speech
   k. google-cloud

7) Install Redis Server on VM

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install redis-server
sudo service redis-server restart
```

Once, the Redis server is installed and running (default port 6379), we can use below command to test if Redis is running. The output will be PONG.

```
redis-cli ping
```

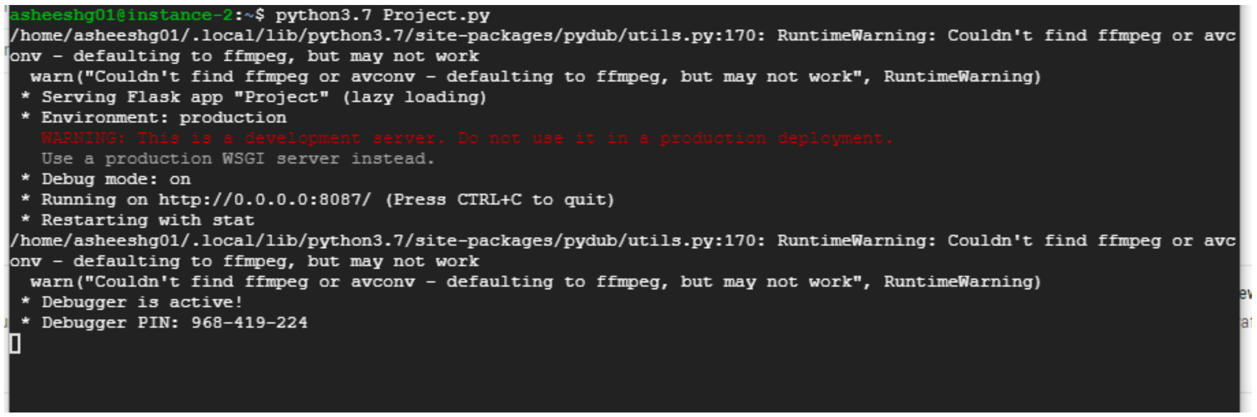
8) Pip Install python 3.7 and Redis in VM

9) Store files Project.py, config.py, Speech-67da50aab51c.json and worker.py in VM home directory. Create Directory under VM home by the name input. All the code has been uploaded in Github at below link

   https://github.com/Asheeshg/Speech-to-text/upload

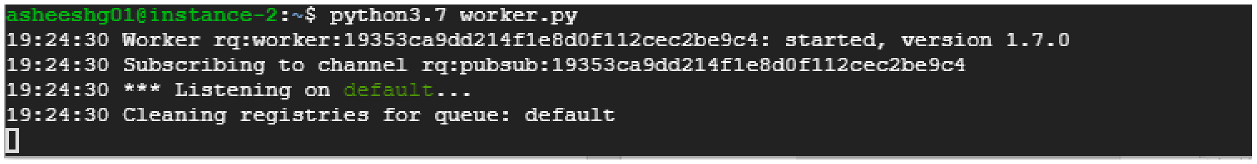
10) Run project app by firing the below command

   Python3.7 Project.py



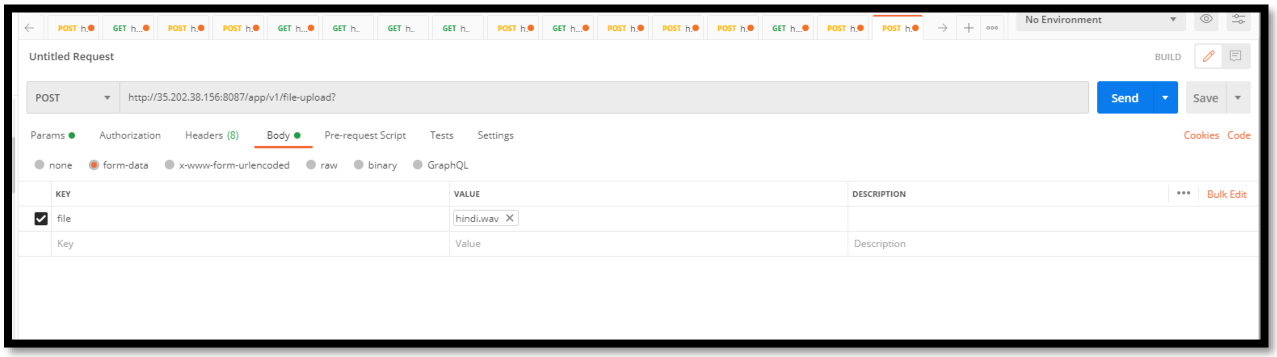
```
asheeshg01@instance-2:~$ python3.7 Project.py
/home/asheeshg01/.local/lib/python3.7/site-packages/pydub/utils.py:170: RuntimeWarning: Couldn't find ffmpeg or avc
onv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
 * Serving Flask app "Project" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8087/ (Press CTRL+C to quit)
 * Restarting with stat
/home/asheeshg01/.local/lib/python3.7/site-packages/pydub/utils.py:170: RuntimeWarning: Couldn't find ffmpeg or avc
onv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
 * Debugger is active!
 * Debugger PIN: 968-419-224
```

11) Fire up below command in another VM terminal window. This will listen for a queue called default and established a connection to Redis server on a localhost:6379



```
asheeshg01@instance-2:~$ python3.7 worker.py
19:24:30 Worker rq:worker:19353ca9dd214f1e8d0f112cec2be9c4: started, version 1.7.0
19:24:30 Subscribing to channel rq:pubsub:19353ca9dd214f1e8d0f112cec2be9c4
19:24:30 *** Listening on default...
19:24:30 Cleaning registries for queue: default
```

12) Through Postman, fire a POST request at http://<external IP of VM>:8087/app/v1/file-upload. In body, please enter file as KEY and select file in KEY cell dropdown. In VALUE cell, click on 'Select Files' button and select the input audio file. Please note that audio file should be in .wav format and should be either English or Hindi.


Untitled Request

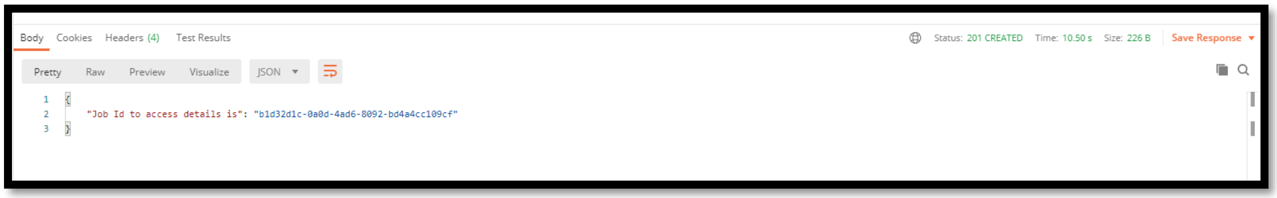
POST http://35.202.38.156:8087/app/v1/file-upload?

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION
✓	file	hindi.wav	
	Key	Value	Description

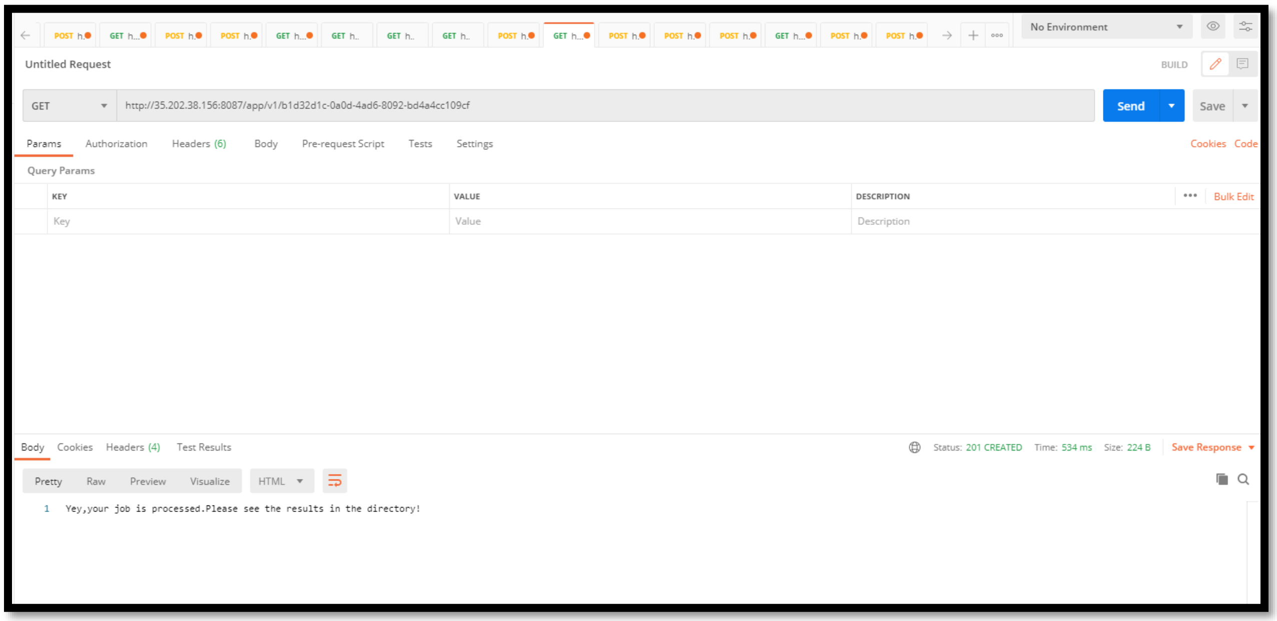
13) Click on Send button in Postman and should be able to see output like below


Body Cookies Headers (4) Test Results — Status: 201 CREATED Time: 10.50 s Size: 226 B

```
{
    "Job Id to access details is": "b1d32d1c-0a0d-4ad6-8092-bd4a4cc109cf"
}
```

As can be seen above, Job id is returned after firing the asynchronous API. Please note down this Job id as this will be used in subsequent steps to check whether Job is completed or not.

14) Please fire the below GET request in postman to retrieve the status of the job
http://<external IP of VM>:8087/app/v1/<job-id>


Untitled Request

GET http://35.202.38.156:8087/app/v1/b1d32d1c-0a0d-4ad6-8092-bd4a4cc109cf

Params Authorization Headers (6) Body Pre-request Script Tests Settings

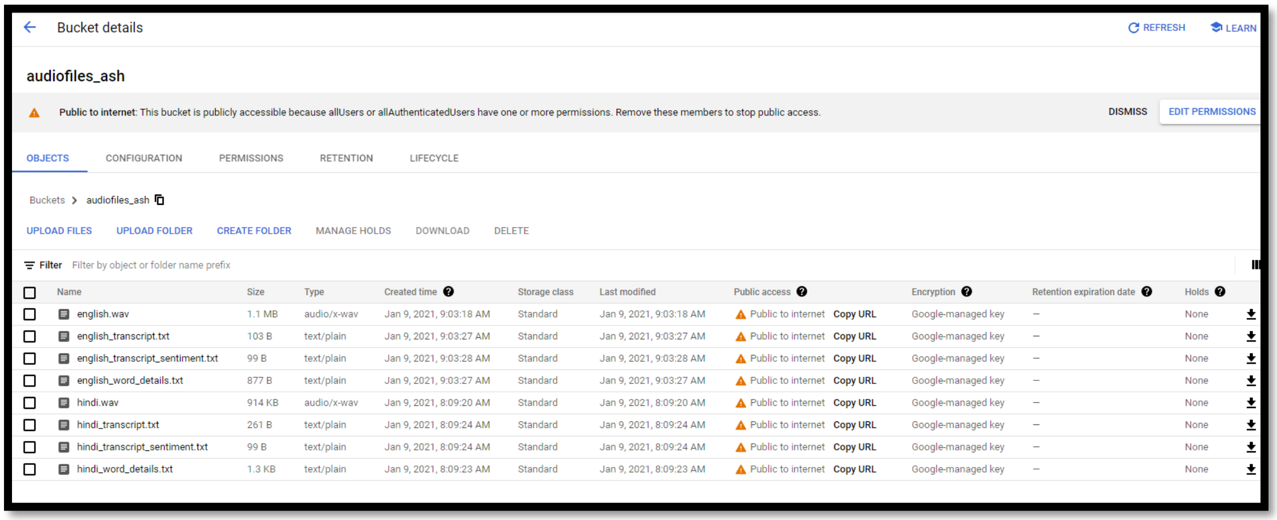
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results — Status: 201 CREATED Time: 534 ms Size: 224 B

```
Yey,your job is processed.Please see the results in the directory!
```

If message received is “Yey, your job is processed. Please see the results in the directory!” then this means that job is completed and all the outputs can be checked in Google Global storage in audiofiles_ash bucket. Please note that bucket name can be changed by modifying the same in json stored in Google Secret Manager.



Bucket details

audiofiles_ash

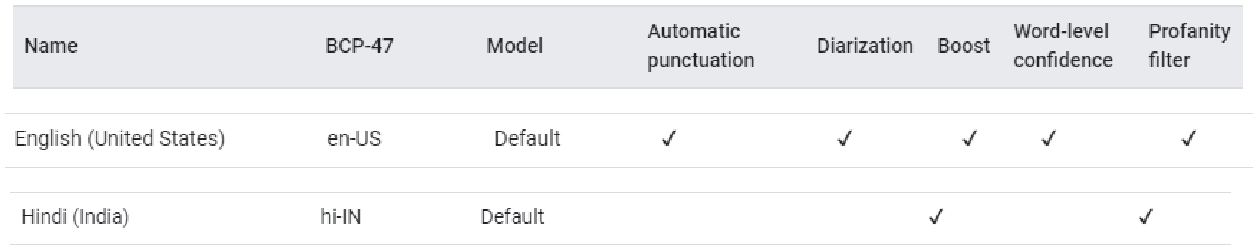
Public to internet: This bucket is publicly accessible because allUsers or allAuthenticatedUsers have one or more permissions. Remove these members to stop public access.

Buckets > audiofiles_ash

Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
english.wav	1.1 MB	audio/x-wav	Jan 9, 2021, 9:03:18 AM	Standard	Jan 9, 2021, 9:03:18 AM	Public to internet Copy URL	Google-managed key	–	None
english_transcript.txt	103 B	text/plain	Jan 9, 2021, 9:03:27 AM	Standard	Jan 9, 2021, 9:03:27 AM	Public to internet Copy URL	Google-managed key	–	None
english_transcript_sentiment.txt	99 B	text/plain	Jan 9, 2021, 9:03:28 AM	Standard	Jan 9, 2021, 9:03:28 AM	Public to internet Copy URL	Google-managed key	–	None
english_word_details.txt	877 B	text/plain	Jan 9, 2021, 9:03:27 AM	Standard	Jan 9, 2021, 9:03:27 AM	Public to internet Copy URL	Google-managed key	–	None
hindi.wav	914 KB	audio/x-wav	Jan 9, 2021, 8:09:20 AM	Standard	Jan 9, 2021, 8:09:20 AM	Public to internet Copy URL	Google-managed key	–	None
hindi_transcript.txt	261 B	text/plain	Jan 9, 2021, 8:09:24 AM	Standard	Jan 9, 2021, 8:09:24 AM	Public to internet Copy URL	Google-managed key	–	None
hindi_transcript_sentiment.txt	99 B	text/plain	Jan 9, 2021, 8:09:24 AM	Standard	Jan 9, 2021, 8:09:24 AM	Public to internet Copy URL	Google-managed key	–	None
hindi_word_details.txt	1.3 KB	text/plain	Jan 9, 2021, 8:09:23 AM	Standard	Jan 9, 2021, 8:09:23 AM	Public to internet Copy URL	Google-managed key	–	None

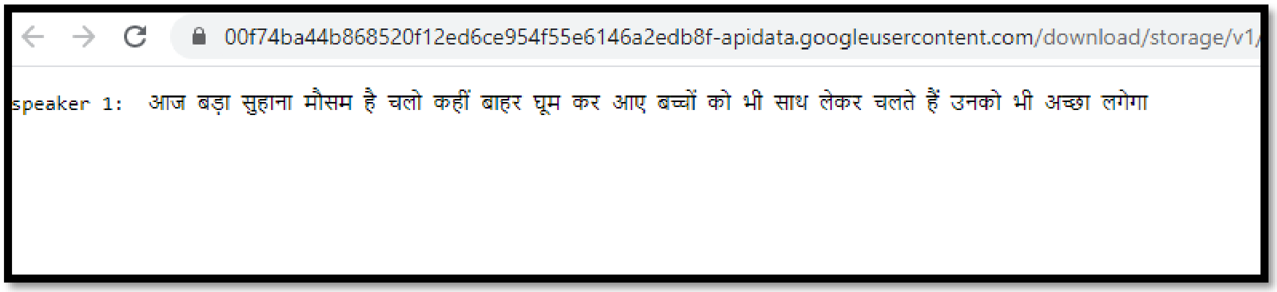
As can be seen above the input audio file hindi.wav has three output files:

a) <audio_filename>_transcript.txt : This file contains the converted audio to txt transcript. Please note that speakers will be distinguished for English language as Diarization is supported by Google Speech-to-Text API, however for Hindi language Diarization is not supported by Google APIs.

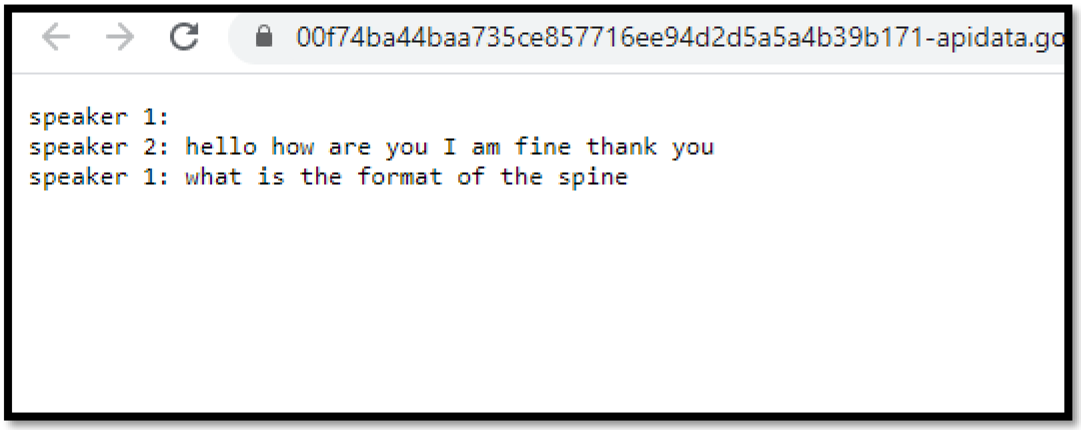


Name	BCP-47	Model	Automatic punctuation	Diarization	Boost	Word-level confidence	Profanity filter
English (United States)	en-US	Default	✓	✓	✓	✓	✓
Hindi (India)	hi-IN	Default			✓		✓

User should be able to see the transcript after downloading the file. Note- Please make sure that UTF-8 is enabled on the browser to see Devanagari fonts.

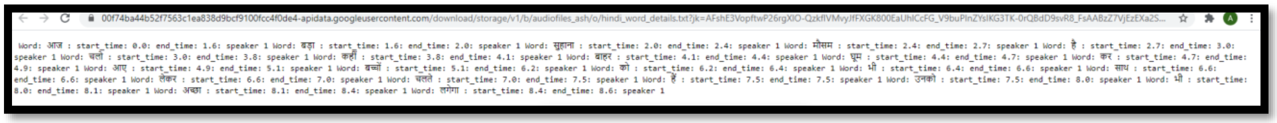

00f74ba44b868520f12ed6ce954f55e6146a2edb8f-apidata.googleusercontent.com/download/storage/v1/

```
speaker 1:  आज बड़ा सुहाना मौसम है चलो कहीं बाहर घूम कर आए बच्चों को भी साथ लेकर चलते हैं उनको भी अच्छा लगेगा
```

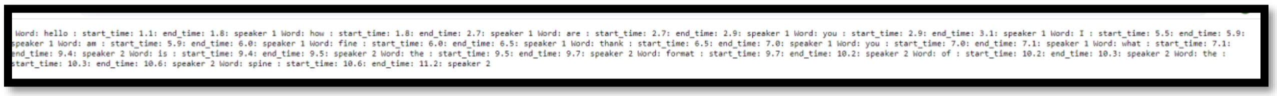


```
speaker 1:
speaker 2: hello how are you I am fine thank you
speaker 1: what is the format of the spine
```

b) <audio_filename>_word_details.txt : This file contains the word details like who said which word at what start time and end time. User should be able to see output like below.


00f74ba44b52f7563c1ea838d9bcf9100fcc4f0de4-apidata.googleusercontent.com/download/storage/v1/b/audiofiles_ash/o/hindi_word_details.txt?jk=AFshE3VopftwP26rgXIO-QzkflVMvyJfFXGK800EaUhlCcFG_V9buPlnZYsIKG3TK-0rQBdD9svR8_FsAABzZ7VjEzEXa2S...

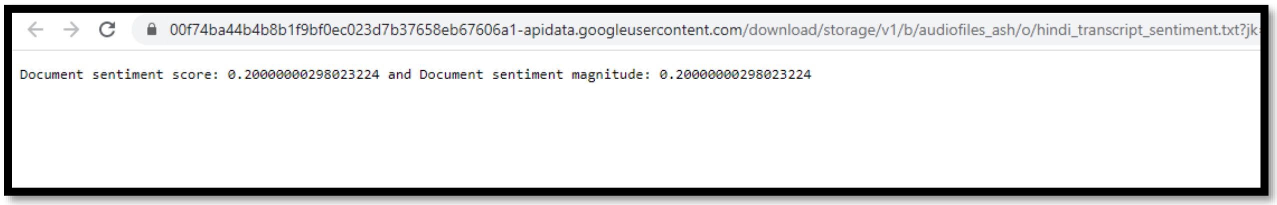
```
Word: आज : start_time: 0.0: end_time: 1.6: speaker 1 Word: बड़ा : start_time: 1.6: end_time: 2.0: speaker 1 Word: सुहाना : start_time: 2.0: end_time: 2.4: speaker 1 Word: मौसम : start_time: 2.4: end_time: 2.7: speaker 1 Word: है : start_time: 2.7: end_time: 3.0:
speaker 1 Word: चलो : start_time: 3.0: end_time: 3.8: speaker 1 Word: कहीं : start_time: 3.8: end_time: 4.1: speaker 1 Word: बाहर : start_time: 4.1: end_time: 4.4: speaker 1 Word: घूम : start_time: 4.4: end_time: 4.7: speaker 1 Word: कर : start_time: 4.7: end_time:
4.9: speaker 1 Word: आए : start_time: 4.9: end_time: 5.1: speaker 1 Word: बच्चों : start_time: 5.1: end_time: 6.2: speaker 1 Word: को : start_time: 6.2: end_time: 6.4: speaker 1 Word: भी : start_time: 6.4: end_time: 6.6: speaker 1 Word: साथ : start_time: 6.6:
end_time: 6.6: speaker 1 Word: लेकर : start_time: 6.6: end_time: 7.0: speaker 1 Word: चलते : start_time: 7.0: end_time: 7.5: speaker 1 Word: हैं : start_time: 7.5: end_time: 7.5: speaker 1 Word: उनको : start_time: 7.5: end_time: 8.0: speaker 1 Word: भी : start_time:
8.0: end_time: 8.1: speaker 1 Word: अच्छा : start_time: 8.1: end_time: 8.4: speaker 1 Word: लगेगा : start_time: 8.4: end_time: 8.6: speaker 1
```



```
Word: hello : start_time: 1.1: end_time: 1.8: speaker 1 Word: how : start_time: 1.8: end_time: 2.7: speaker 1 Word: are : start_time: 2.7: end_time: 2.9: speaker 1 Word: you : start_time: 2.9: end_time: 3.1: speaker 1 Word: I : start_time: 5.5: end_time: 5.9:
speaker 1 Word: am : start_time: 5.9: end_time: 6.0: speaker 1 Word: fine : start_time: 6.0: end_time: 6.5: speaker 1 Word: thank : start_time: 6.5: end_time: 7.0: speaker 1 Word: you : start_time: 7.0: end_time: 7.1: speaker 1 Word: what : start_time: 7.1:
end_time: 9.4: speaker 2 Word: is : start_time: 9.4: end_time: 9.5: speaker 2 Word: the : start_time: 9.5: end_time: 9.7: speaker 2 Word: format : start_time: 9.7: end_time: 10.2: speaker 2 Word: of : start_time: 10.2: end_time: 10.3: speaker 2 Word: the :
start_time: 10.3: end_time: 10.6: speaker 2 Word: spine : start_time: 10.6: end_time: 11.2: speaker 2
```

c) <audio_filename>_transcript_sentiment.txt : This file will provide the sentiment score and magnitude of the overall transcript. The *score* of a document's sentiment indicates the overall emotion of a document. The *magnitude* of a document's sentiment indicates how much emotional content is present within the document, and this value is often proportional to the length of the document. Please see details on how to interpret sentiment score and magnitude at below link.

https://cloud.google.com/natural-language/docs/basics#interpreting_sentiment_analysis_values



00f74ba44b4b8b1f9bf0ec023d7b37658eb67606a1-apidata.googleusercontent.com/download/storage/v1/b/audiofiles_ash/o/hindi_transcript_sentiment.txt?jk

```
Document sentiment score: 0.20000000298023224 and Document sentiment magnitude: 0.20000000298023224
```

Will load docker image and instructions in GIthub later.