Setup Instructions

This project processes audio files using speaker diarization (PyAnnote) and word-level transcription (AssemblyAI), then generates timestamped CSV files for each recording.

Prerequisites

* AWS account with access to Lambda and IAM
* Your own `AssemblyAI` and `PyAnnote` API keys
* Public S3 URLs or signed links to audio files

Step-by-Step Instructions

Step1: Create a S3 bucket

1. Go to the S3 Console and create a new bucket (e.g., ratestaudio).
2. Upload your audio files (.wav, .m4a).
3. Open the **Permissions** tab → Enable **Public Access**.
4. In the **Bucket Policy**, paste the following content to allow public read access to the audio files:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublicReadGetObject",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:GetObject",
   "Resource": "arn:aws:s3:::ratestaudio/*"
  }
```

]

}

Step 2: Prepare Python Dependencies (Locally)

```bash

mkdir python

pip install assemblyai requests -t python/

zip -r layer.zip python

```

Step 3: Upload Layer

1. Go to **AWS Lambda → Layers

2. Click Create layer

3. Upload `layer.zip`

4. Compatible runtimes: `Python 3.9` (or whatever your function uses)

5. After creation, attach the layer to your Lambda function

Step 4: Create Your Lambda Function

1. Runtime: Python 3.9

2. Upload `lambda_function.py`

3. Set handler as:

```

lambda_function.lambda_handler

```

4. Assign an IAM Role to your Lambda function, and ensure the role includes the following permissions:

- Allow access to S3 (if your Lambda is triggered by S3 or needs to read files from S3)
- Allow outbound internet access (via NAT Gateway or similar) to reach the PyAnnote and AssemblyAI APIs

Step 5: Set Environment Variables

In the Lambda console → Configuration → Environment variables:

* `ASSEMBLYAI_API_KEY`

* `PYANNOTE_API_KEY`

Step 6: Set Timeout

Since diarization + transcription takes time, increase timeout:

* Go to Lambda → Configuration → General configuration

* Set timeout to at least `2 minutes`

Step 7: Test Lambda

1. Use a test event like:

```json
{}
```

2. The Lambda will:

  * Diarize speakers using PyAnnote

  * Transcribe audio using AssemblyAI

  * Return a base64-encoded .zip of all CSVs

Step 8: Run in Browser

1. Enable **Function URL** for your Lambda function:

    - Go to your function → "Function URL" tab
    - Copy the generated URL (e.g. https://xyz123.lambda-url.us-east-1.on.aws/)
2. Open this URL in a browser
3. Your browser will automatically download a .zip file
   — containing the CSV results for all audio files

Output

Each CSV contains:

| speaker | start | end | text |

| ------- | ----- | --- | ---- |

Speakers will be labeled as `parent` or `child` based on who spoke first.

Sample Layer Directory Layout:

```
layer.zip
└──── python/
    ├──── assemblyai/
    ├──── requests/
    └──── ...
```