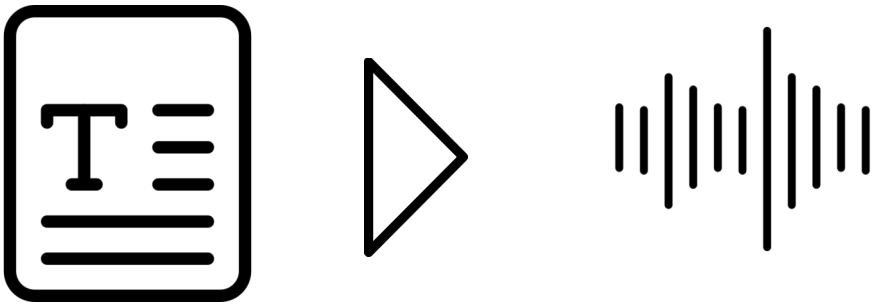
TEXT-TO-SPEECH AWS PROJECT



February 2025

Objectives

The goal of this project was to automate the conversion of text files into speech using AWS services. When a .txt file is uploaded to a specific folder in an S3 bucket, an AWS Lambda function is triggered. This function processes the text file using Amazon Polly to generate an .mp3 file, which is then stored in a designated output folder within the same bucket.

Application Architecture

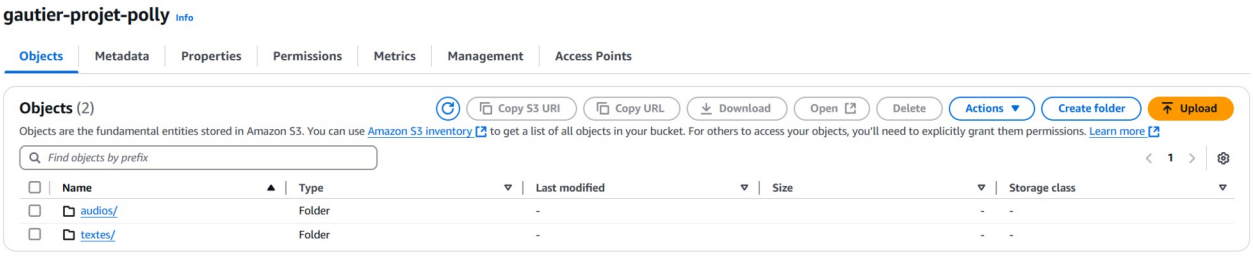
The solution was implemented using the following AWS services:

- **Amazon S3** – Used as the storage solution for input text files and output audio files.
- **AWS Lambda** – Processes the text file, converts it to speech, and stores the resulting audio file.
- **Amazon Polly** – Performs the text-to-speech conversion.

Amazon S3

I created a bucket *gautier-projet-polly* with two sub folders:
- */textes – Stores the uploaded .txt files.*
- */audios – Stores the generated .mp3 files.*

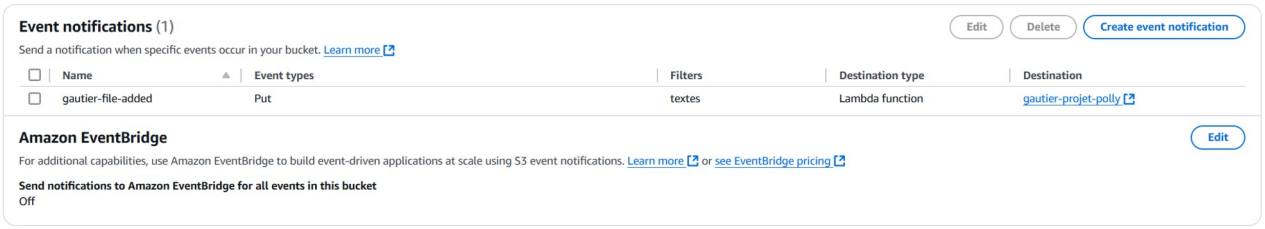
The *textes* folder is, as the name implies, made for the .txt files and the other folder is where the audio files will be stored.



No special bucket policies were required for this implementation.

I then configured S3 to trigger a Lambda function whenever a file is uploaded to the /textes folder.

We can do that without having to use Amazon EventBridge, directly in the *Properties* tab on the S3 bucket as described below:



AWS Lambda

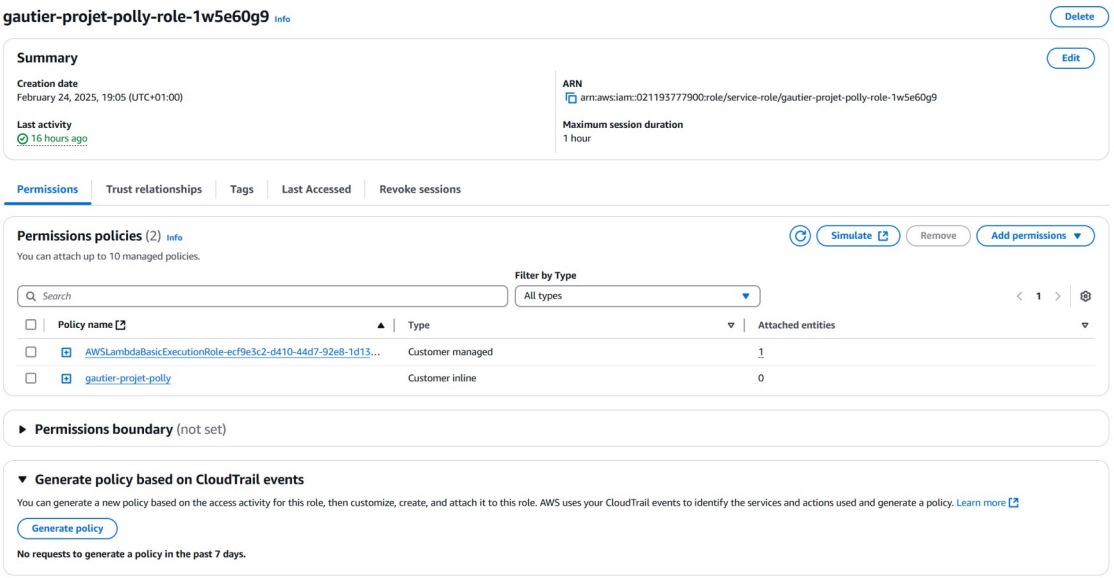
The Lambda function was written in Python and performs the following steps:

1. **Triggered by an S3 event** when a new .txt file is uploaded.
2. **Reads the content** of the text file.
3. **Uses Amazon Polly** to convert the text into an audio file.
4. **Saves the generated .mp3 file** in the */audio* folder, maintaining the original filename.

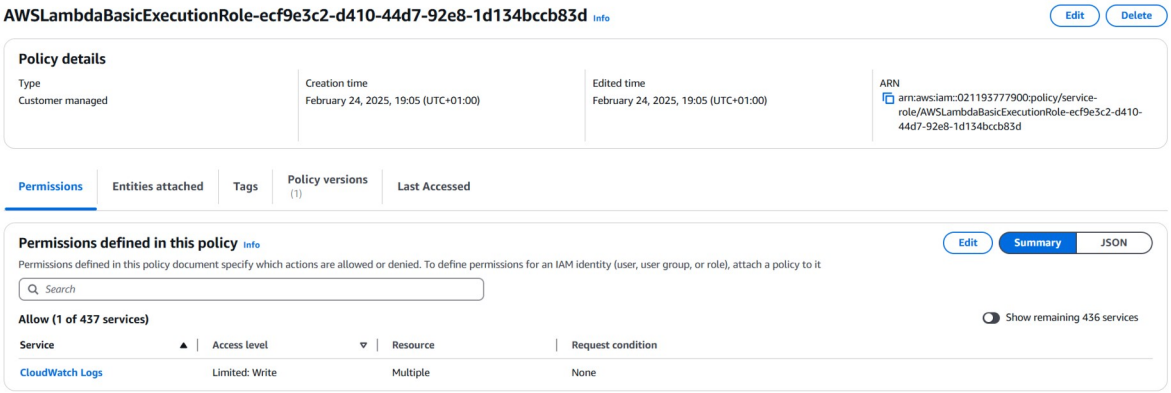
```python
import json
import boto3
import os
from contextlib import closing

def lambda_handler(event, context):

    file = event["Records"][0]["s3"]["object"]["key"] #get the file name

    bucket_name = event["Records"][0]["s3"]["bucket"]["name"] #get the bucket name

    s3 = boto3.client('s3')

    polly_client = boto3.client('polly')

    text = s3.get_object(Bucket=bucket_name, Key=file)['Body'].read().decode('utf-8')

    file_out = file.split('/')[-1] #removes the prefix (textes/)

    file_out = file_out.lower().replace(' ', '_').replace('(', '').replace(')', '').replace('.txt', '')
    file_out = file_out + '.mp3'

    response = polly_client.synthesize_speech(VoiceId='Joanna',
                                              LanguageCode='en-US',
                                              OutputFormat='mp3',
                                              TextType='text',
                                              Text = text)

    # Access the audio stream from the response
    if "AudioStream" in response:
        # Note: Closing the stream is important because the service throttles on the
        # number of parallel connections. Here we are using contextlib.closing to
        # ensure the close method of the stream object will be called automatically
        # at the end of the with statement's scope.
        with closing(response["AudioStream"]) as stream:
            output = os.path.join('/tmp', file_out)
            try:
                # Open a file for writing the output as a binary stream
                with open(output, "wb") as file:
                    file.write(stream.read())
            except IOError as error:
                # Could not write to file, exit gracefully
                print(error)
                sys.exit(-1)
    else:
        # The response didn't contain audio data, exit gracefully
        print("Could not stream audio")
        sys.exit(-1)

    s3.upload_file('/tmp/' + file_out, bucket_name,  'audios/' + file_out )

    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

Required Permissions:

- Write access to **CloudWatch Logs** for debugging and monitoring.
- Read access to **S3 */textes* folder** and write access to S3 */audio* folder.
- Access to Amazon Polly to perform text-to-speech conversion.



- Right to write logs in CloudWatch:



- Access to Amazon Polly, S3 (read in */textes* and write in */audio*):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "polly:*",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::gautier-projet-polly/*",
                "arn:aws:s3:::gautier-projet-polly"
            ]
        }
    ]
}
```

Amazon Polly

Amazon Polly was integrated into the **AWS Lambda** function to perform **text-to-speech (TTS) conversion**. The function reads the content of the uploaded .txt file and uses **Polly's neural voices** to generate an .mp3 file. This audio file is then stored in the **/audios** folder in Amazon S3.

Problems faced and solutions

Lambda function Timeout :

- The default execution timeout for AWS Lambda is **3 seconds**, which was insufficient for processing longer text files.
- Observing logs in **CloudWatch**, I noticed the function would restart repeatedly when processing longer files.
- Solution: Increased the timeout to **3 minutes**, ensuring sufficient time for the function to complete execution.
- After adjustment, I monitored execution times in **AWS Lambda Monitor Menu**, confirming that files with multiple sentences were processed in approximately **5 seconds**.