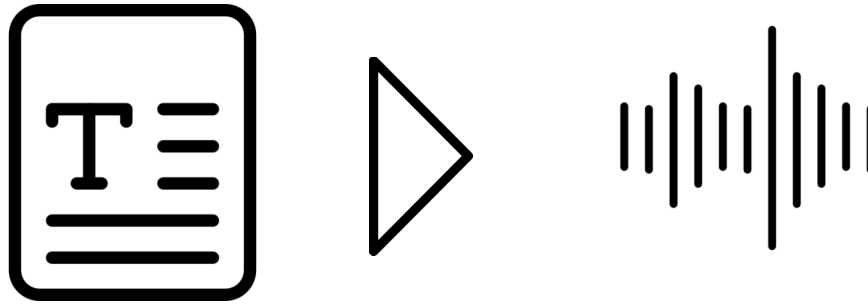


TEXT-TO-SPEECH AWS PROJECT



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:

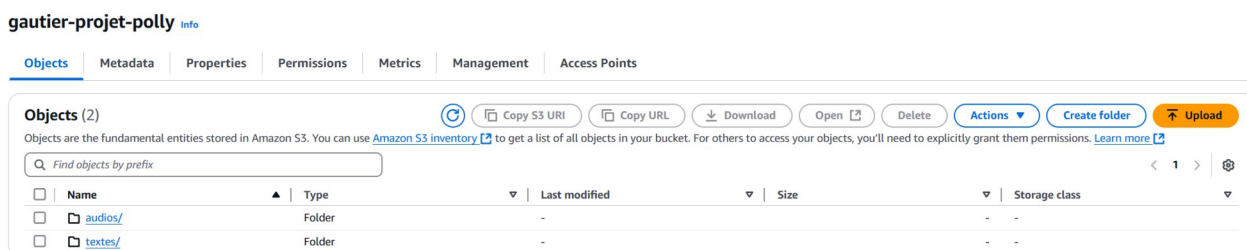
- **Amazon S3** – Used as the storage solution for input text files and output audio files.
- **AWS Lambda** – Processes the text file, converts it to speech, and stores the resulting audio file.
- **Amazon Polly** – Performs the text-to-speech conversion.

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:

Event notifications (1)

Edit
Delete
Create event notification

☐ Name

☐ Event types

☐ Filters

☐ Destination type

☐ Destination

☐ gautier-file-added

☐ Put

☐ textes

☐ Lambda function

☐ gautier-projet-polly

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications.
Learn more
or see EventBridge pricing

Send notifications to Amazon EventBridge for all events in this bucket
Off

Edit

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.

```

1 import json
2 import boto3
3 import os
4 from contextlib import closing
5
6 def lambda_handler(event, context):
7
8     file = event["Records"][0]["s3"]["object"]["key"] #get the file name
9
10    bucket_name = event["Records"][0]["s3"]["bucket"]["name"] #get the bucket name
11
12    s3 = boto3.client('s3')
13
14    polly_client = boto3.client('polly')
15
16    text = s3.get_object(Bucket=bucket_name, Key=file)['Body'].read().decode('utf-8')
17
18    file_out = file.split('/')[1] #removes the prefix (textes/)
19
20    file_out = file_out.lower().replace(' ', '_').replace('(', '').replace(')', '').replace('.txt', '')
21    file_out = file_out + '.mp3'
22
23    response = polly_client.synthesize_speech(VoiceId='Joanna',
24                                             LanguageCode='en-US',
25                                             OutputFormat='mp3',
26                                             TextType='text',
27                                             Text = text)
28
29    # Access the audio stream from the response
30    if "AudioStream" in response:
31        # Note: Closing the stream is important because the service throttles on the
32        # number of parallel connections. Here we are using contextlib.closing to
33        # ensure the close method of the stream object will be called automatically
34        # at the end of the with statement's scope.
35        with closing(response["AudioStream"]) as stream:
36            output = os.path.join('/tmp', file_out)
37            try:
38                # Open a file for writing the output as a binary stream
39                with open(output, "wb") as file:
40                    file.write(stream.read())
41            except IOError as error:
42                # Could not write to file, exit gracefully
43                print(error)
44                sys.exit(-1)
45    else:
46        # The response didn't contain audio data, exit gracefully
47        print("Could not stream audio")
48        sys.exit(-1)
49
50    s3.upload_file('/tmp/' + file_out, bucket_name, 'audios/' + file_out )
51
52    return {
53        'statusCode': 200,
54        'body': json.dumps('Hello from Lambda!')
55    }
56
57
58

```

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.

gautier-projet-polly-role-1w5e60g9

Summary

Creation date
February 24, 2025, 19:05 (UTC+01:00)

Last activity
16 hours ago

ARN
arn:aws:iam::021193777900:role/service-role/gautier-projet-polly-role-1w5e60g9

Maximum session duration
1 hour

Permissions

Trust relationships

Tags

Last Accessed

Revoke sessions

Permissions policies (2)

You can attach up to 10 managed policies.

Search

Filter by Type
All types

< 1 >

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole-ecf9e3c2-d410-44d7-92e8-1d13...	Customer managed	1
gautier-projet-polly	Customer inline	0

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Generate policy

No requests to generate a policy in the past 7 days.

- Right to write logs in CloudWatch:

AWSLambdaBasicExecutionRole-ecf9e3c2-d410-44d7-92e8-1d134bccb83d

Policy details

Type
Customer managed

Creation time
February 24, 2025, 19:05 (UTC+01:00)

Edited time
February 24, 2025, 19:05 (UTC+01:00)

ARN
arn:aws:iam::021193777900:policy/service-role/AWSLambdaBasicExecutionRole-ecf9e3c2-d410-44d7-92e8-1d134bccb83d

Permissions

Entities attached

Tags

Policy versions (1)

Last Accessed

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 437 services)

Show remaining 436 services

Service	Access level	Resource	Request condition
CloudWatch Logs	Limited: Write	Multiple	None

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

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "polly:*",
8       "Resource": "*"
9     },
10    {
11      "Sid": "VisualEditor1",
12      "Effect": "Allow",
13      "Action": [
14        "s3:PutObject",
15        "s3:GetObject",
16        "s3:ListBucket"
17      ],
18      "Resource": [
19        "arn:aws:s3:::gautier-projet-polly/*",
20        "arn:aws:s3:::gautier-projet-polly"
21      ]
22    }
23  ]
24 }
```

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**.

General configuration Info			Edit
Description	Memory	Ephemeral storage	
-	128 MB	512 MB	
Timeout	SnapStart Info		
3 min 0 sec	None		