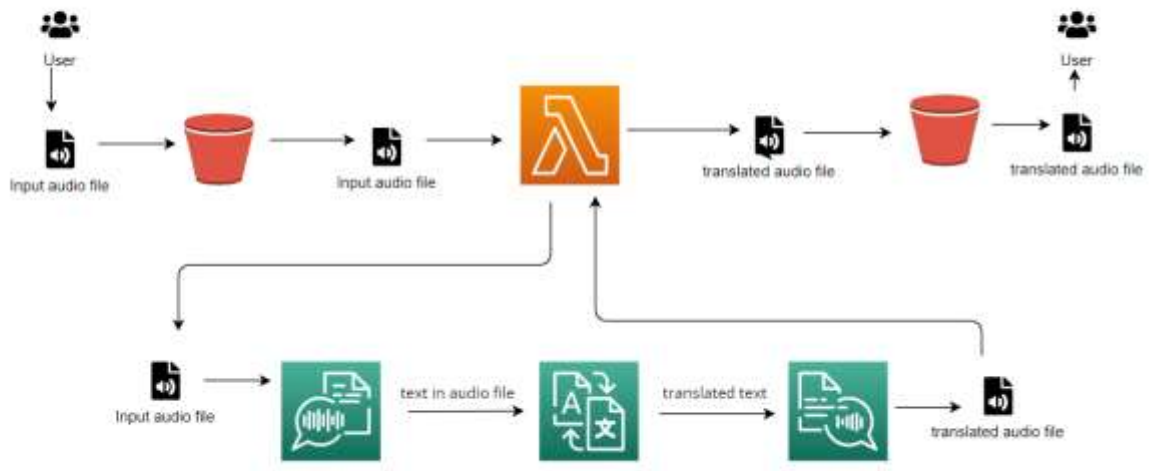
Cloud Audio Translator: Seamless Multilingual Communication with AWS

Introduction:

This project presents a cloud-based solution designed to translate audio files from one language to another automatically. Utilizing AWS Lambda functions triggered by S3 events, the system seamlessly converts uploaded audio files into translated versions, enhancing accessibility and cross-language communication.

Project Overview:



Event Trigger:

The process begins when a user uploads an audio file to a specified S3 bucket. This action triggers an event that initiates the translation process.

Transcription and Translation:

Transcription: Upon receiving the S3 event, an AWS Lambda function is invoked to transcribe the audio file using Amazon Transcribe. This converts the spoken content into text, allowing for further processing.

Translation: The transcribed text is then translated into the desired language (supported by AWS) using Amazon Translate. This step ensures that the content is comprehensible to users who speak different languages.

Text-to-Speech Conversion:

Segmentation: To avoid exceeding text length limitations, the translated text is segmented into smaller chunks.

Synthesis: Each translated text segment undergoes text-to-speech synthesis using Amazon Polly. This generates an audio file containing the translated speech in the target language.

Output Generation:

The translated audio segments are concatenated to form the final translated audio file.

The resulting audio file is stored in a designated S3 bucket, making it accessible to users for playback or download.

Implementation Details:

AWS Services used:

- AWS Lambda
- Amazon S3
- Amazon Transcribe
- Amazon Translate
- Amazon Poll

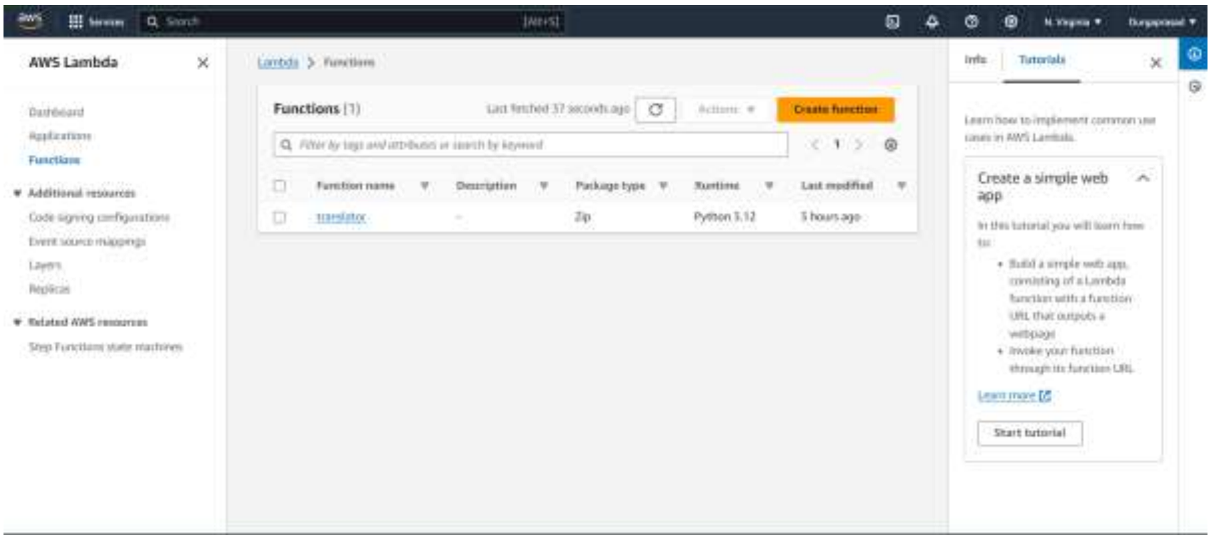
Creating Lambda Function:

Sign in to AWS Console: Log in to the AWS Management Console.

Navigate to Lambda: Go to the Lambda service by typing "Lambda" in the search bar or selecting it from the list of services.

Create Function:

Click on the "Create function" button.



AWS Lambda

Lambda > Functions

Functions (1)

Last fetched 37 seconds ago

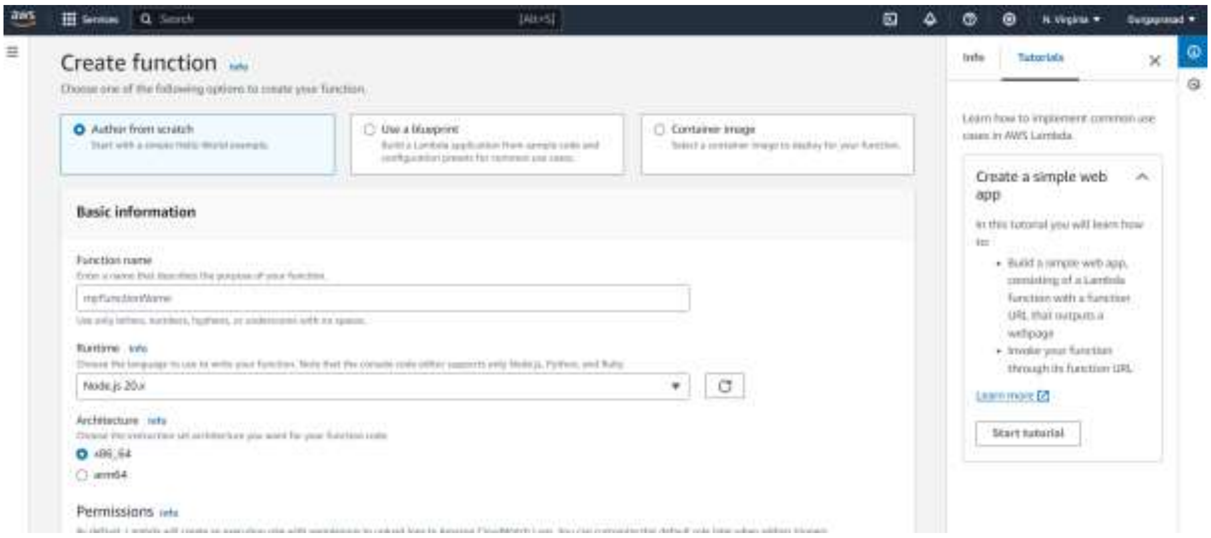
Function name	Description	Package type	Runtime	Last modified
translator	-	Zip	Python 3.12	5 hours ago

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Choose "Author from scratch" as the option.



Create function

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint

Container image

Basic information

Function name: myFunctionName

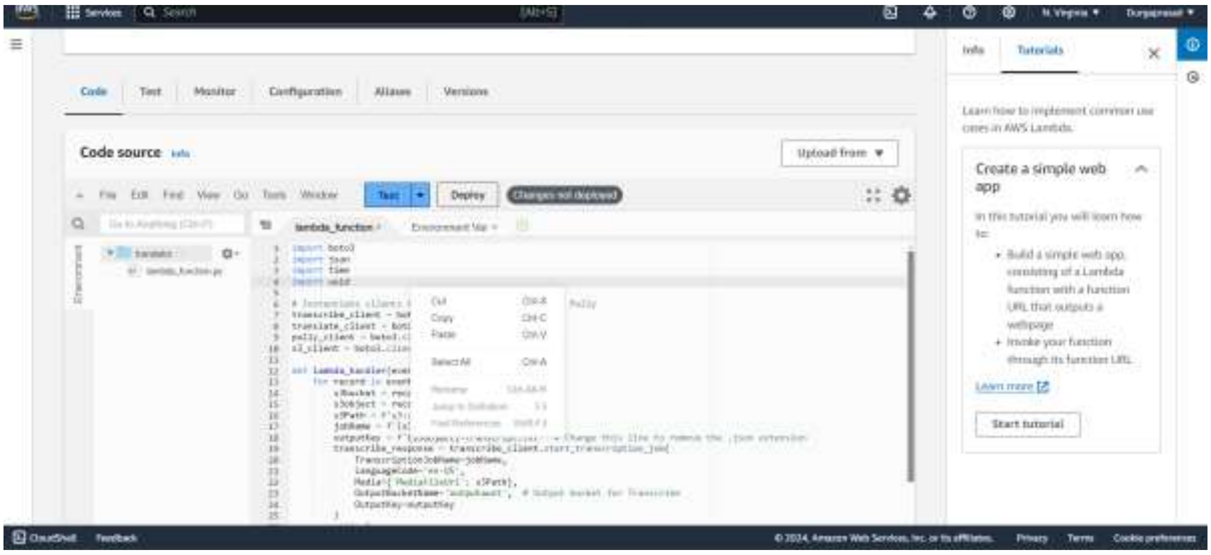
Runtime: Node.js 20.x

Architecture: [✓ x86_64] arm64

Permissions

Provide a name for your function, select the runtime (e.g., Python), and choose an existing role or create a new one with necessary permissions.

Write Code: In the function code section, paste the code for your Lambda function. This code should include the logic to transcribe, translate, and synthesize audio as per your project requirements.



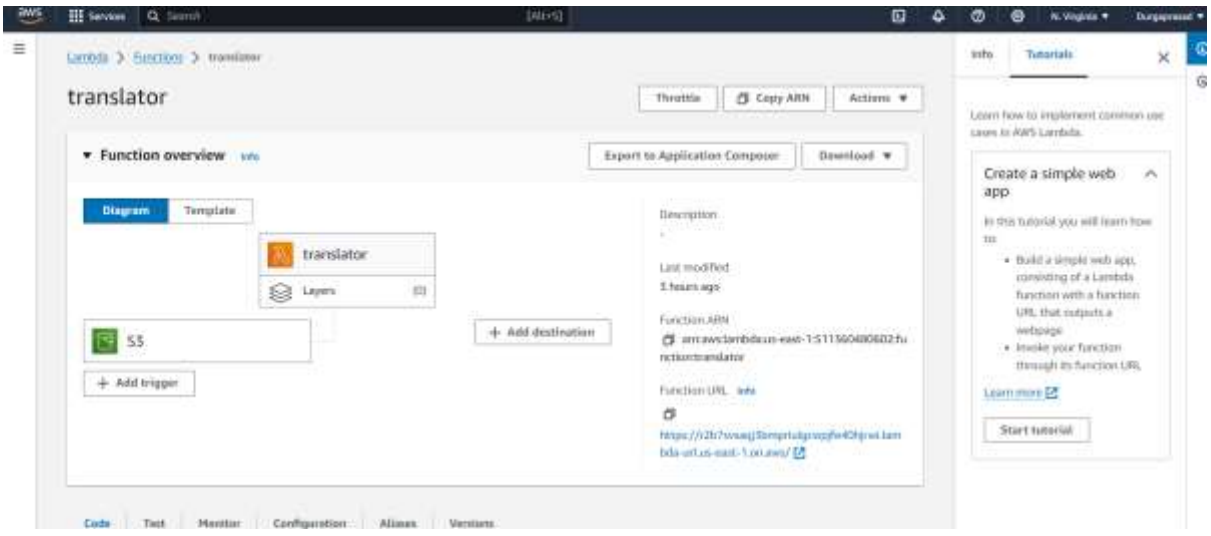
Lambda code:

```
import boto3
import json
import time
import uuid

# Instantiate clients for Transcribe, Translate, and Polly
transcribe_client = boto3.client('transcribe')
translate_client = boto3.client('translate')
polly_client = boto3.client('polly')
s3_client = boto3.client('s3')

def lambda_handler(event, context):
    for record in event['Records']:
        s3bucket = record['s3']['bucket']['name']
        s3object = record['s3']['object']['key']
        s3Path = f's3://{s3bucket}/{s3object}'
        jobName = f'{s3object}--{str(uuid.uuid4())}'
        outputKey = f'{s3object}-transcript.txt'  # Change this line to remove the .json extension
        transcribe_response = transcribe_client.start_transcription_job(
            TranscriptionJobName=jobName,
            LanguageCode='en-US',
            Media={'MediaFileUri': s3Path},
            OutputBucketName='outputawst',  # Output bucket for Transcribe
            OutputKey=outputKey
        )
        while True:
            status = transcribe_client.get_transcription_job(TranscriptionJobName=jobName)
            if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
                break
            time.sleep(10)
        # Get the transcription result
        transcription_result = s3_client.get_object(Bucket='outputawst', Key=outputKey)
        transcript_text = json.load(transcription_result['Body'])['results']['transcripts'][0]['transcript']
        # Translate the transcript
        transcript_text = str(transcript_text)
        l = len(transcript_text)
```

```
transcribeoutput = [transcript_text[:l//2], transcript_text[l//2:]]
translated_text = ''
for i in transcribeoutput:
    translate_response = translate_client.translate_text(
        Text=i,
        SourceLanguageCode='en',
        TargetLanguageCode='hi'  # Specify your target language code here, e.g., 'fr' for French
    )
    translated_text += translate_response['TranslatedText']

# Split translated text into chunks to avoid TextLengthExceededException
chunk_size = 500  # Adjust chunk size as needed
translated_chunks = [translated_text[i:i+chunk_size] for i in range(0, len(translated_text), chunk_size)]

# Convert translated text chunks to speech using Polly
output_audio = b''
for chunk in translated_chunks:
    response = polly_client.synthesize_speech(
        Text=chunk,
        OutputFormat='mp3',
        VoiceId='Aditi'  # Choose the desired Polly voice
    )
    output_audio += response['AudioStream'].read()

# Store the output audio file in the "outputawstt" bucket
output_audio_key = f'{s3object}-translated-audio.mp3'
s3_client.put_object(Body=output_audio, Bucket='outputawstt', Key=output_audio_key)
```

Configure Trigger:

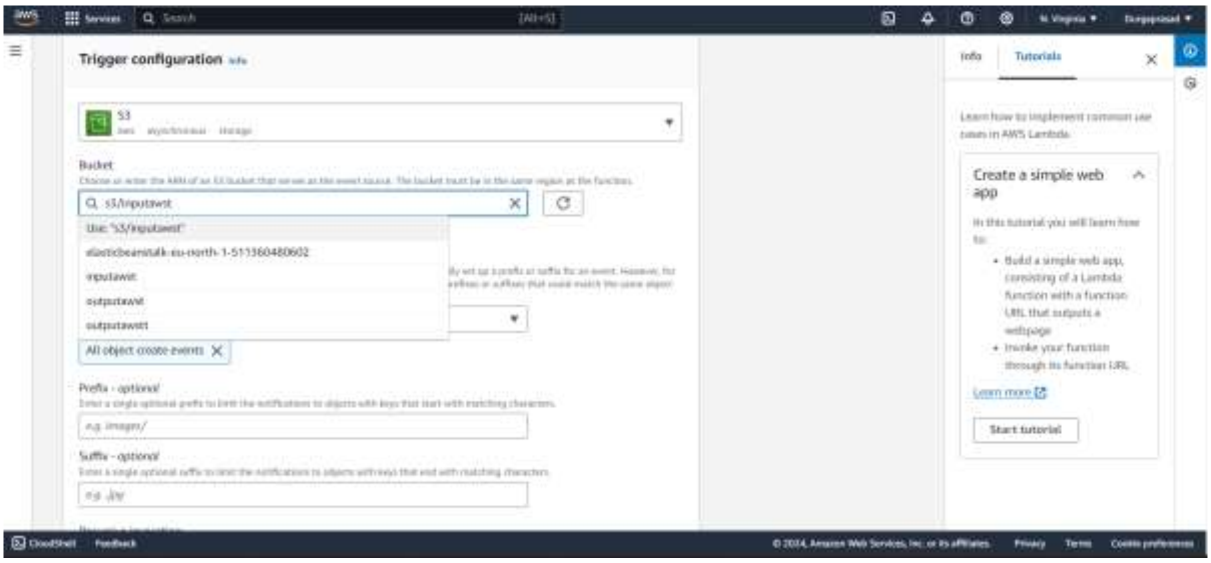
Scroll down to the "Add triggers" section and select "S3" from the list of trigger options.



Setting Up S3 Trigger:

Choose Trigger Source:

Select the bucket where users will upload audio files.



Trigger configuration

S3

Bucket

s3/inputawst

Use: "s3/inputawst"

elasticbeanstalk-eu-north-1-511360480602

inputawst

outputawst

outputawstt

All object create events

Prefix - optional

e.g. images/

Suffix - optional

e.g. .jpg

Choose the event type that triggers the Lambda function (e.g., "ObjectCreated" for any new object uploaded to the bucket).

Configure Trigger Settings:

Set the prefix and suffix filters if you want to trigger the function only for specific files based on their names.

Optionally, enable "Enable trigger" to activate the trigger immediately.

Save Configuration: Click on the "Add" button to save the trigger configuration.

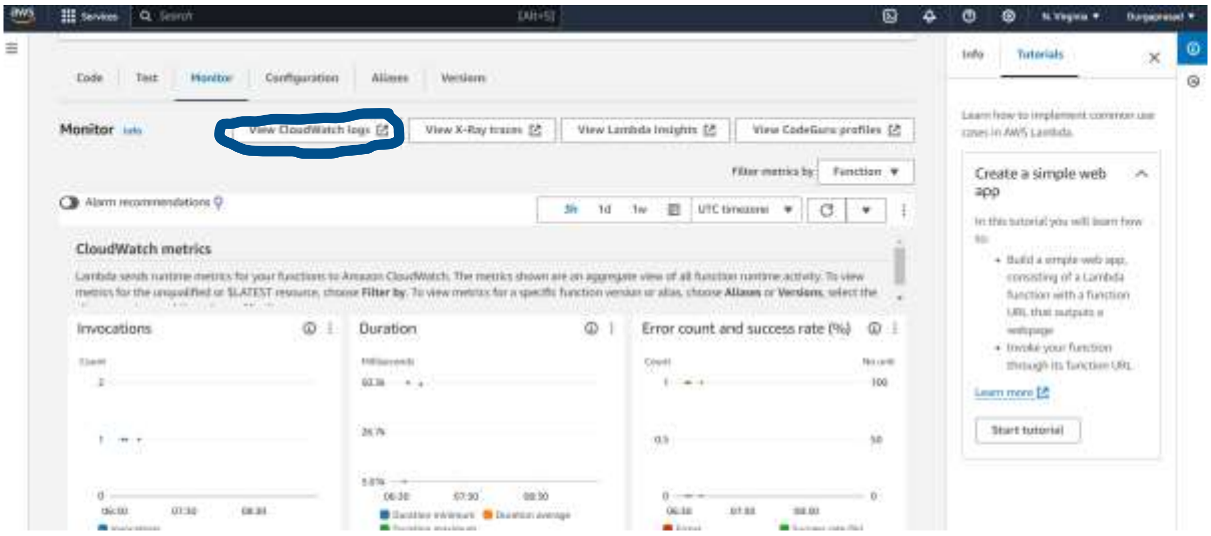
Deploy Lambda Function: Review the function configuration and click on the "Deploy" button to deploy the Lambda function.

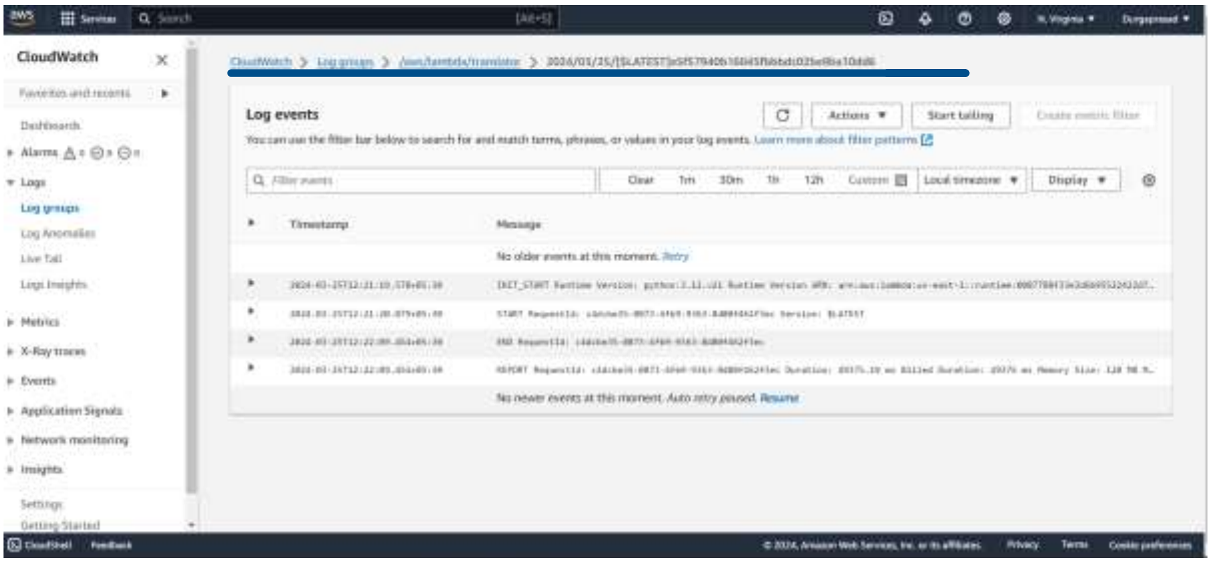
Test the Function: Upload a sample audio file to the specified S3 bucket to test if the Lambda function triggers correctly and performs the desired actions.

Additional Configuration :

Set up logging and monitoring for the Lambda function to track its execution and performance.

Configure error handling and retry mechanisms to handle failures gracefully.





Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2024-03-25T12:21:19.[illegible]  INIT_START Runtime Version: python:3.11.v21 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:[illegible]
2024-03-25T12:21:20.[illegible]  START RequestId: [illegible] Version: $LATEST
2024-03-25T12:22:09.[illegible]  END RequestId: [illegible]
2024-03-25T12:22:09.[illegible]  REPORT RequestId: [illegible] Duration: [illegible] ms Billed Duration: [illegible] ms Memory Size: 128 MB M...
                                 No newer events at this moment. Auto retry paused. Resume
```

Conclusion:

By harnessing the power of cloud computing and AWS services, this project demonstrates a practical and scalable solution for multilingual audio translation. The automated workflow seamlessly converts uploaded audio files into translated versions, facilitating cross-language communication and enhancing accessibility for users worldwide.