# About

In this project we'll be developing a text narrator using Amazon Polly. Any text you want to use, whether a book, article or newsletter, will be uploaded in an Amazon S3 bucket and converted to speech. Additionally, the narrator's voice, pitch and speed parameters can be adjusted to your liking.

The steps we'll be going through to utilize Amazon Polly consist of:

1. Exploring Amazon Polly
2. Creating an IAM role
3. Creating an S3 bucket
4. Writing Lambda function code
5. Checking the output of Amazon Polly

## Services Used

Amazon Polly- Converts text to speech with many customizable features

AWS Management Console- Manages accounts, used to configure Amazon Polly

AWS IAM- Ensures secure access by managing user permissions

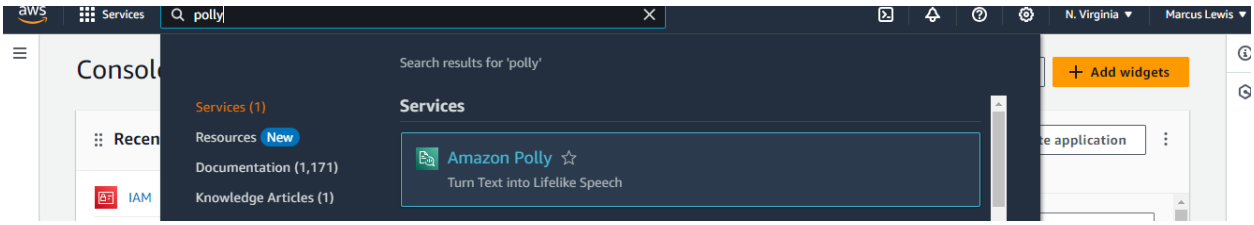
## Time & Cost

This project is estimated to take about 20-30 minutes, and the cost is free with AWS's Free Tier plan.

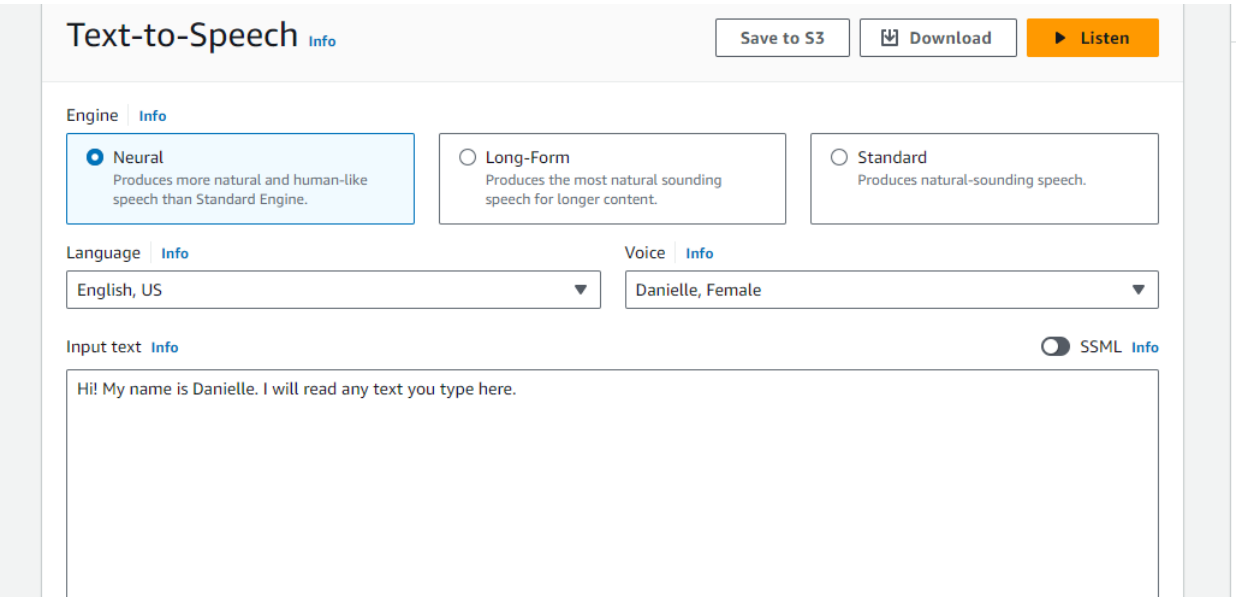
## Step 1

Amazon Polly's selling point is that it allows you to generate human like speech from text, which allows for a more personable experience instead of a robotic automated voice. You can change the pitch, make the voice faster/slower and even use different languages and accents.

To start, we'll login to the AWS Management Console and search for Amazon Polly.




Amazon Polly provides different engines according to the needs, along with a variety of speakers per language.



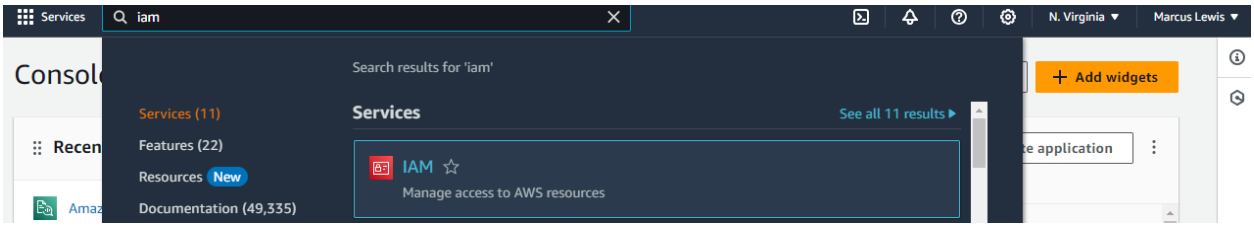

You also have the option to listen to the audio using the options above, download the audio file and save the file to an S3 bucket.

What we'll be accomplishing in this project is learning how to call the Polly service using a serverless function.

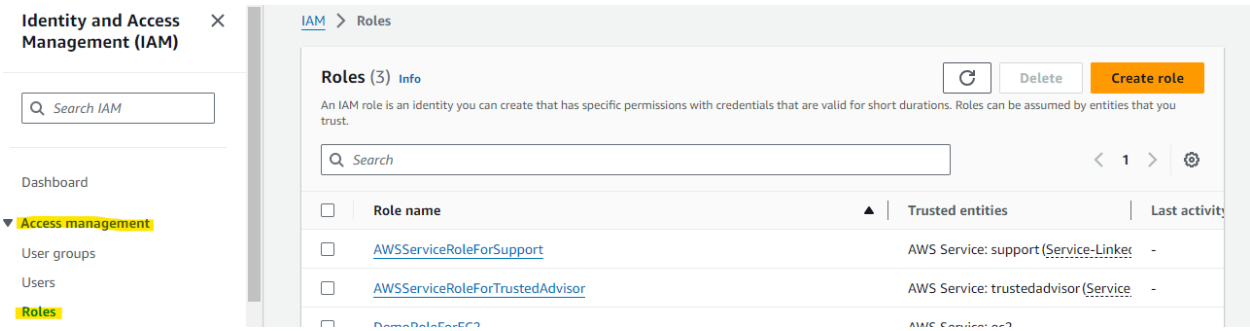
## Step 2

Since our goal is to access Amazon Polly and store the audio output in an S3 bucket, we will need an IAM role with the correct policies attached to it.

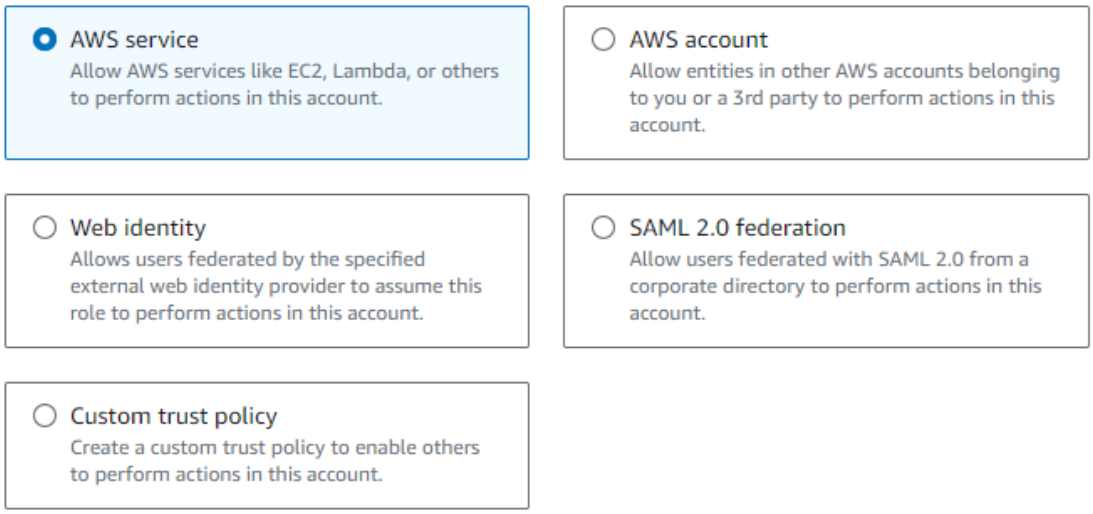
Search for IAM in the AWS management console search bar.

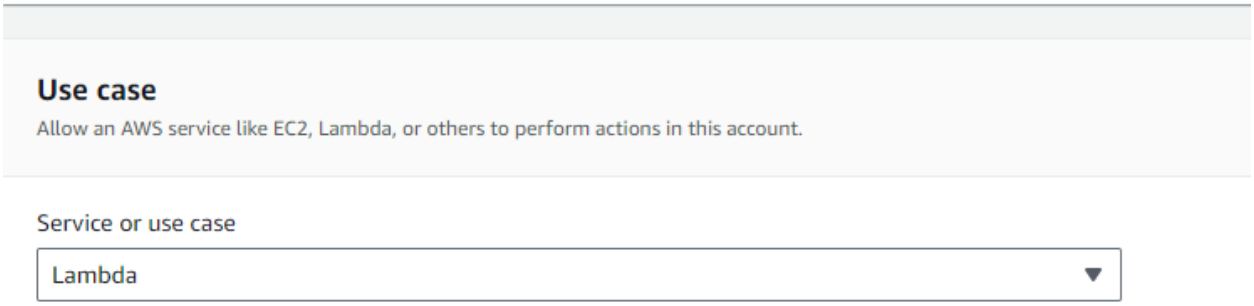



Next, go to Access Management > Roles > Create Role



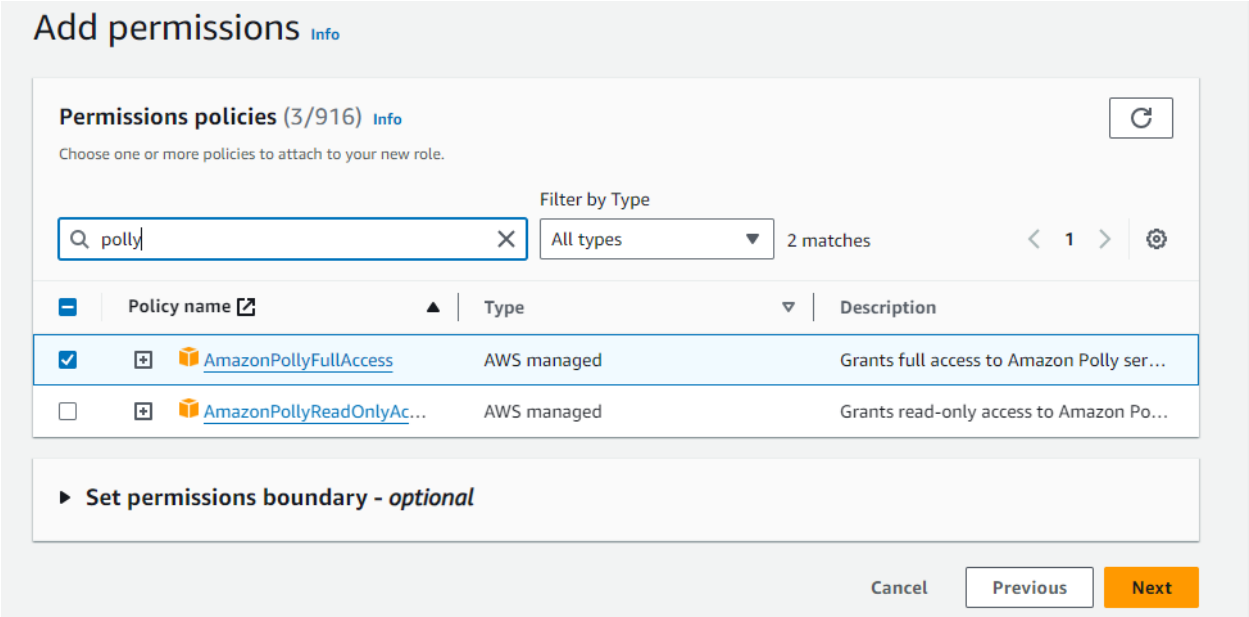

For our trusted entity, keep AWS Service selected and choose Lambda as the use case and click next.



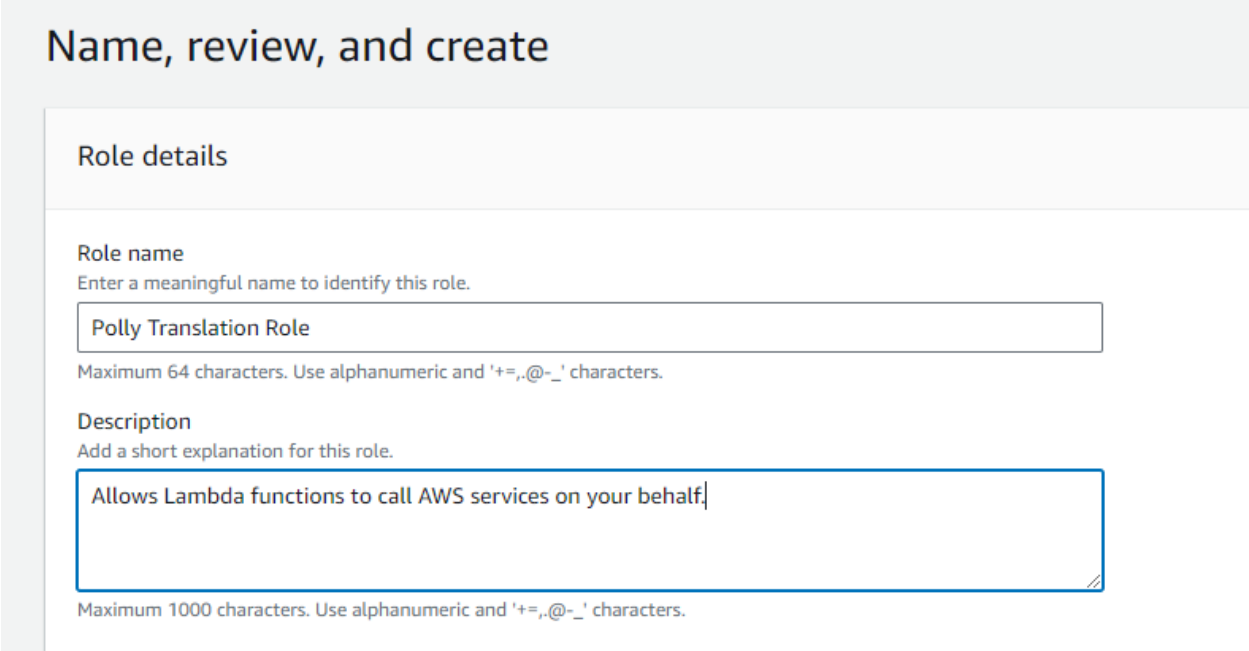




From the list of permission policies on the next screen, choose:

- AmazonPollyFullAccess
- AmazonS3FullAccess
- AWSLambdaBasicExecutionRole

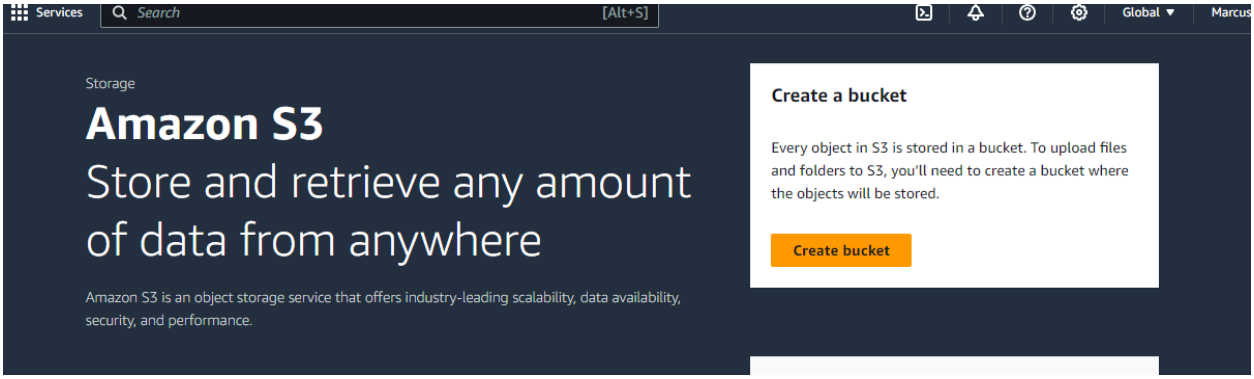



Create a role name and description and click Create Role

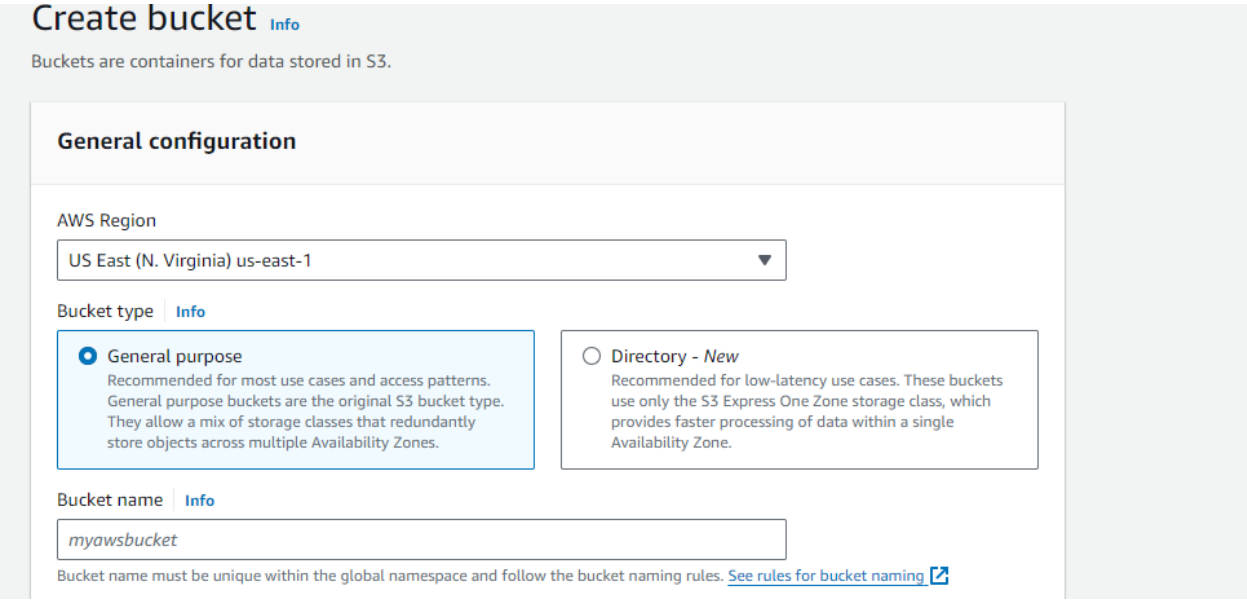



**Step 3**

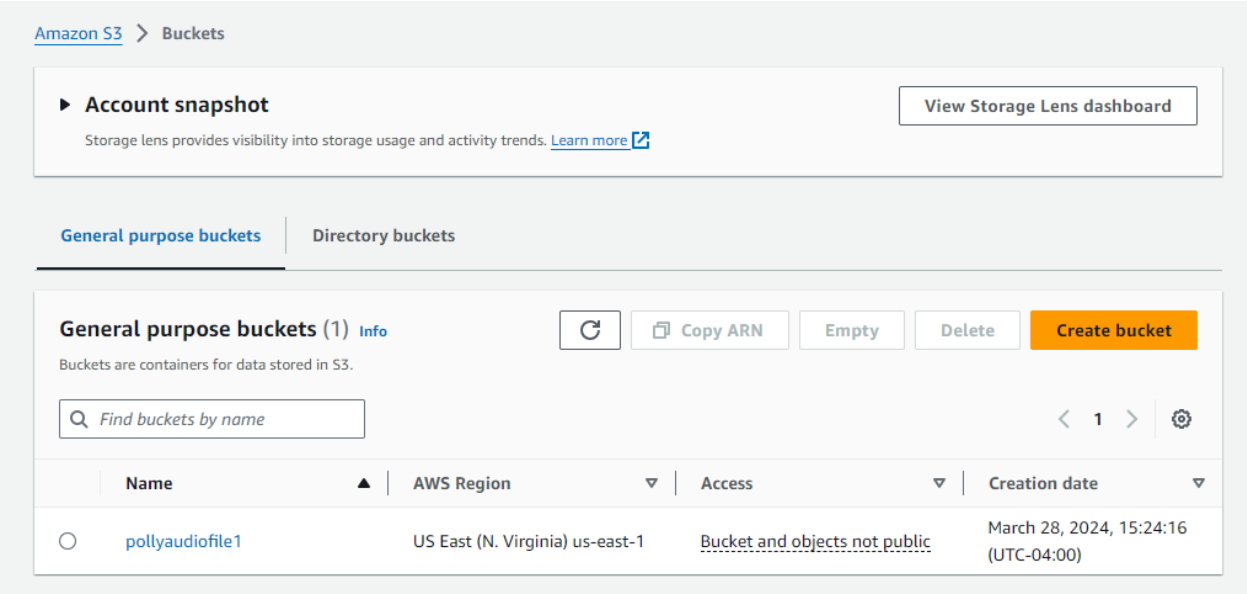
Now that we're done creating the IAM role, search for S3 in the search bar and click Create bucket.




Choose a name for your bucket and leave all other parameters as their default and Create bucket

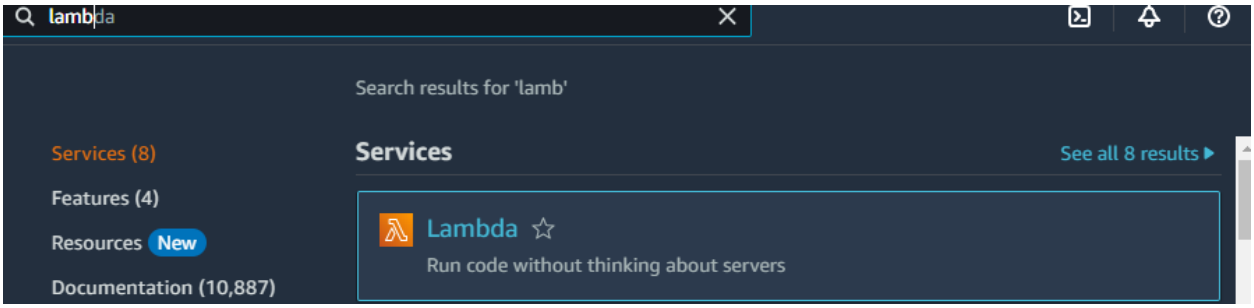



Here we have our created bucket. The audio files we'll be generating from Amazon Polly will be stored in this bucket using AWS Lambda in the next step.

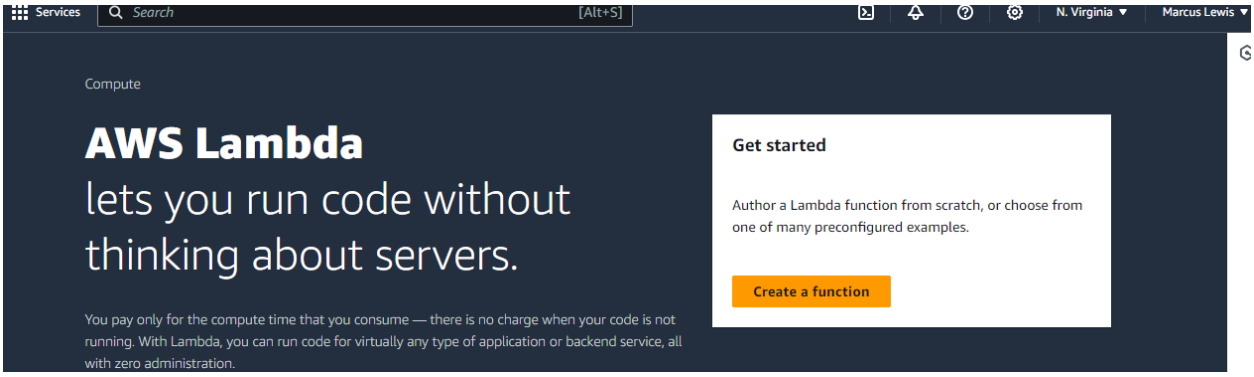



## Step 4

From the management console, type Lambda in the search bar.

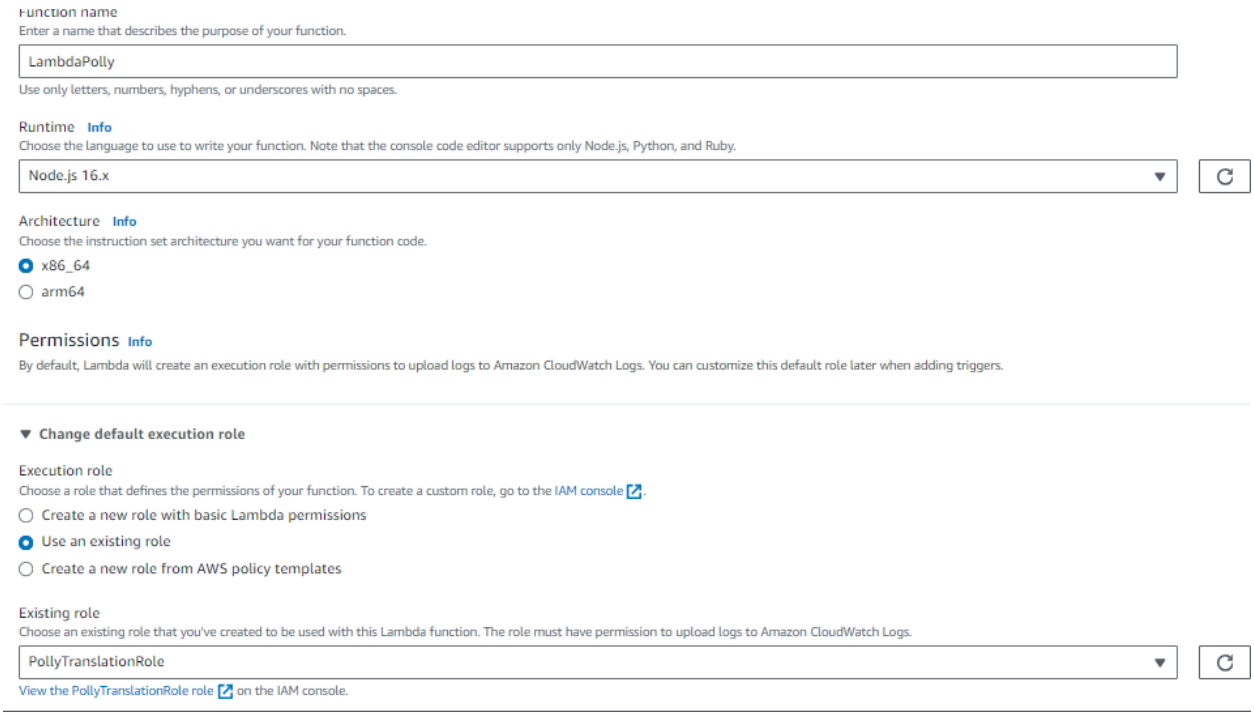



Now click Create a function.



Give our function a name, change the runtime to ‘Node.js 16.x’ as the runtime environment and toggle ‘Change default execution role’ to Use an existing role. The existing role will be the one we created in step 2.

Leave all other configurations as default and click Create function




In the code source section, rename index.mjs to index.js.

We’re going to be using Amazon SDK to talk to two services: Polly for making speech from text and S3 for storing files.

```javascript
const AWS = require('aws-sdk');

const polly = new AWS.Polly();
const s3 = new AWS.S3();
```

Now we write a function that AWS will run for us whenever something happens, like a program that waits for a signal to start working

```javascript
exports.handler = async (event) => {
```

When the function gets a message with some text, we’re going to make it into speech. We decide how the speech will sounds and what format it should be in

```javascript
const text = event.text;

const params = {
    Text: text,
    OutputFormat: 'mp3',
    VoiceId: 'Joanna' // You can change this to the desired voice
};
```

We send the text to Polly and ask it to turn it into speech, which Polly send back to us as a data file that we save in our S3 bucket. Remember to use the actual name of the bucket you created in step 3.

```javascript
const data = await polly.synthesizeSpeech(params).promise();

const key = `audio-${Date.now()}.mp3`;

const s3Params = {
    Bucket: 'your-bucket-name', // Replace with your S3 bucket name
    Key: key,
    Body: data.AudioStream,
    ContentType: 'audio/mpeg'
};

await s3.upload(s3Params).promise();
```

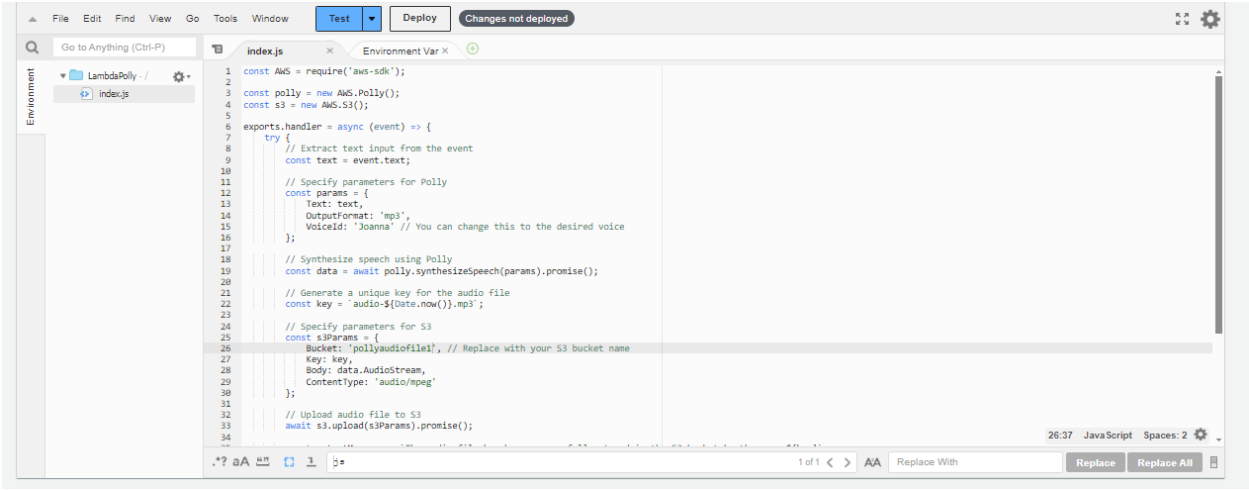
In case of errors, we receive a message telling us the speech has been saved successfully with its special name in our storage. If something goes wrong, we get an error message.

```javascript
const outputMessage = `The audio file has been successfully stored in the S3 bucket by the name ${key}`;

return {
    statusCode: 200,
    body: JSON.stringify({ message: outputMessage })
};

} catch (error) {
    console.error('Error:', error);
    return {
        statusCode: 500,
        body: JSON.stringify({ message: 'Internal server error' })
    };
}
```

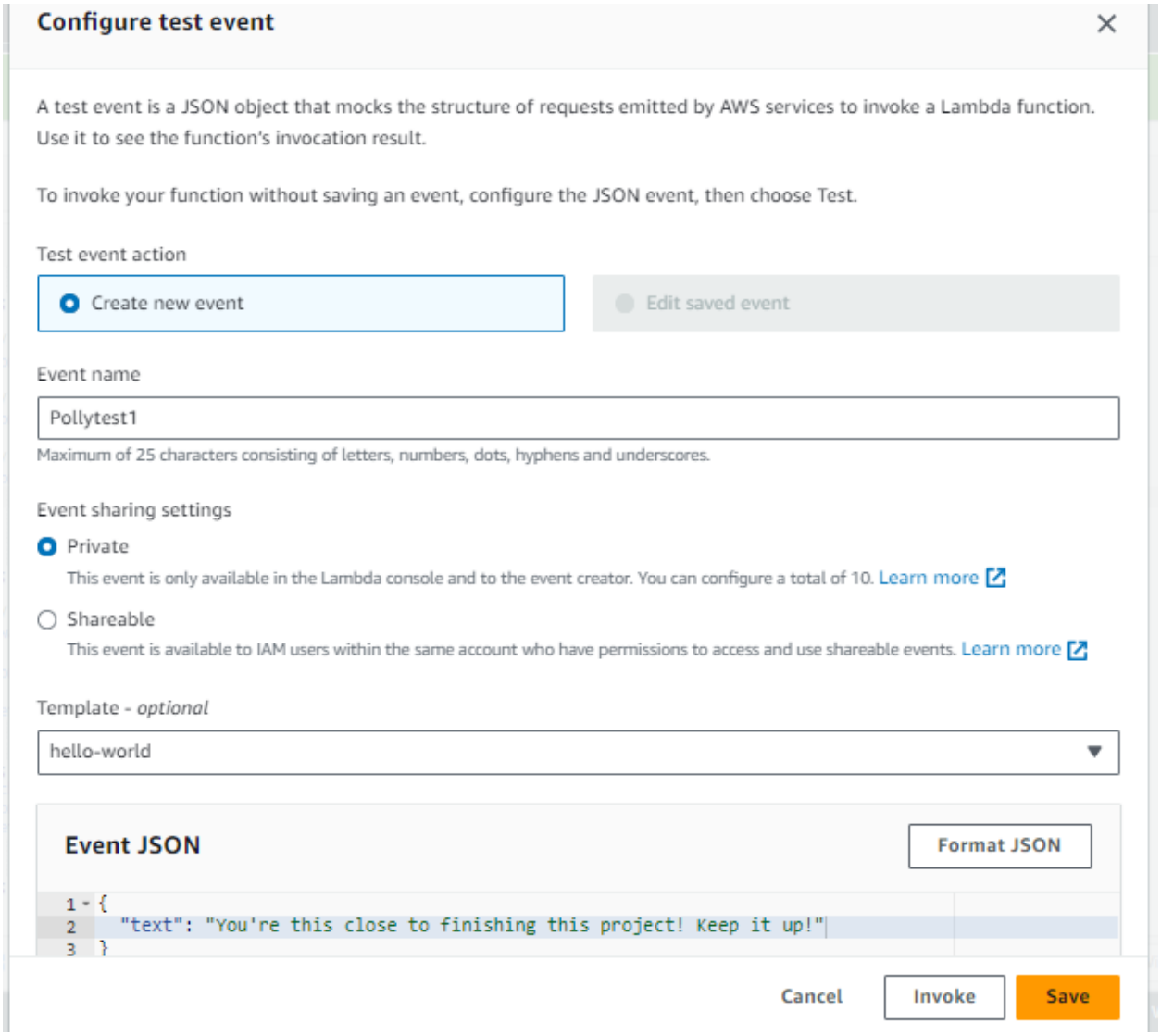
Deploy the code changes we've made by clicking on Deploy



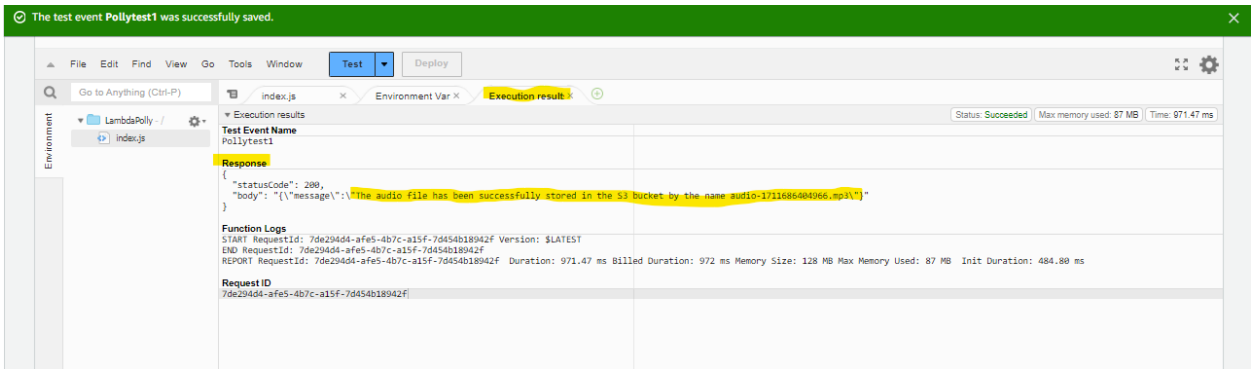

## Step 5

Now that we've completed the code configuration of the Lambda function, let's test the function by creating a test event.

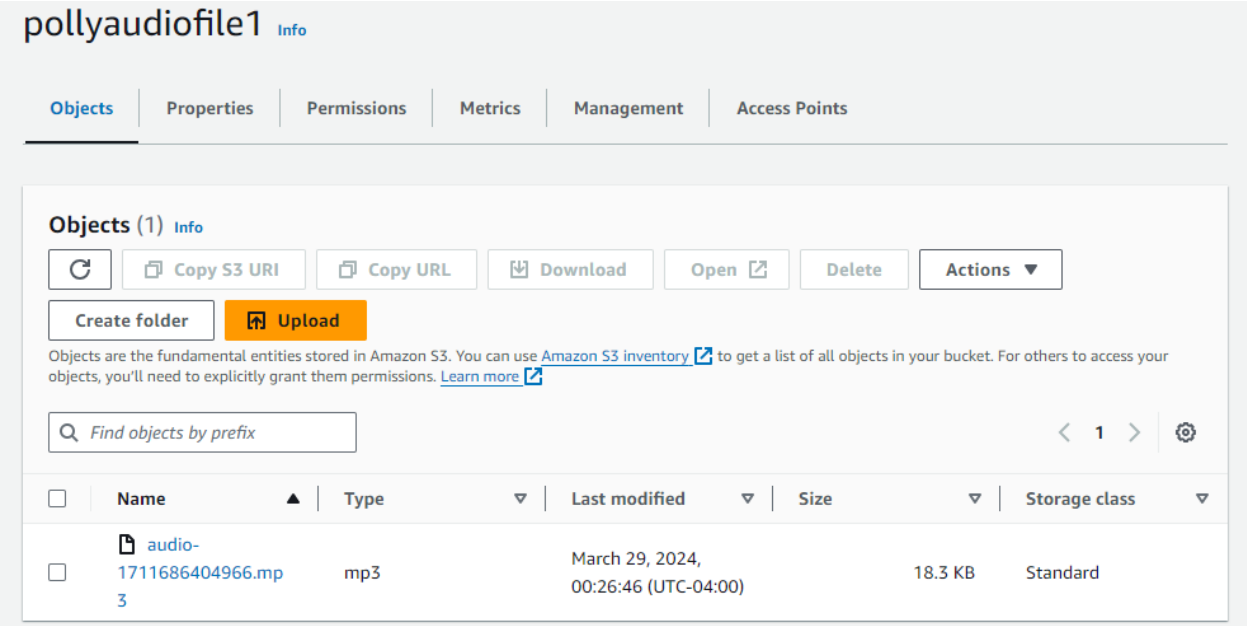
Click on Test; provide a name for our test configuration and in the Event JSON, provide the text you want to be converted to audio. Leave all other configurations as default and click Save.




Now that we've saved, click on Test again to invoke the test event. A tab called Execution Result was created, and the in the response, it




We can now access this audio file by checking it in the previously created S3 bucket




Once you see your created file, you've successfully completed this project! If you've followed along, this is a method of using AWS to create text to speech file using Amazon Polly, Lambda and an S3 bucket.

## Clean Up

From the AWS Management Console, go to S3 and delete the bucket we created for this project. Remember to delete the contents of the bucket first, as you otherwise cannot delete the bucket itself if there's any files remaining.

Do the same for our Lambda function by heading to Lamba from the Management Console and deleting it.

Finally head to IAM and delete the role we created for the Lambda functions.