

## **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

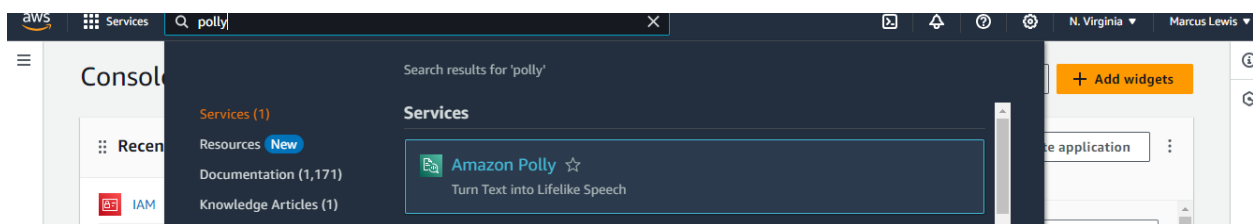
## **Time & Cost**

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.

The screenshot shows the Amazon Polly Text-to-Speech console. At the top, there are buttons for 'Save to S3', 'Download', and 'Listen'. Below these, the 'Engine' section has three radio button options: 'Neural' (selected), 'Long-Form', and 'Standard'. The 'Language' dropdown is set to 'English, US' and the 'Voice' dropdown is set to 'Danielle, Female'. The 'Input text' field contains the text: 'Hi! My name is Danielle. I will read any text you type here.' There is also a checkbox for 'SSML' which is currently unchecked.

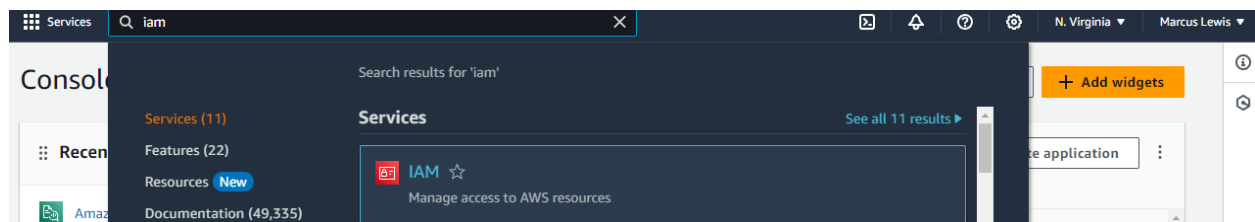
You also have the option to listen to the audio using the options above, download the audio file and save the file to an S3 bucket.

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

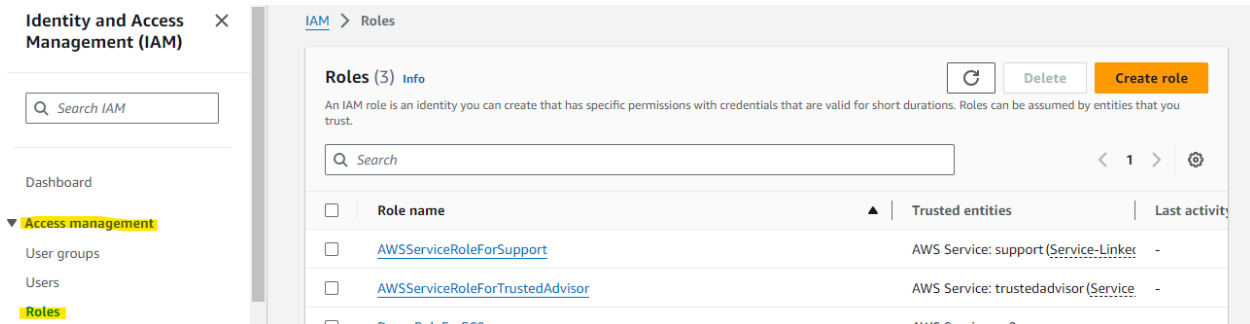
## Step 2

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.

☒ **AWS service**  
 Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**  
 Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**  
 Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**  
 Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**  
 Create a custom trust policy to enable others to perform actions in this account.

---

### Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case  
 Lambda

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



- AmazonPollyFullAccess
- AmazonS3FullAccess
- AWSLambdaBasicExecutionRole

## Add permissions [Info](#)

**Permissions policies (3/916)** [Info](#) ↻

Choose one or more policies to attach to your new role.

× Filter by Type All types ▼ 2 matches < 1 > ⚙

<input type="checkbox"/>	Policy name <a href="#">↗</a>	Type	Description
<input checked="" type="checkbox"/>	 <a href="#">AmazonPollyFullAccess</a>	AWS managed	Grants full access to Amazon Polly ser...
<input type="checkbox"/>	 <a href="#">AmazonPollyReadOnlyAc...</a>	AWS managed	Grants read-only access to Amazon Po...

**▶ Set permissions boundary - optional**

Cancel Previous Next

Create a role name and description and click Create Role

## Name, review, and create

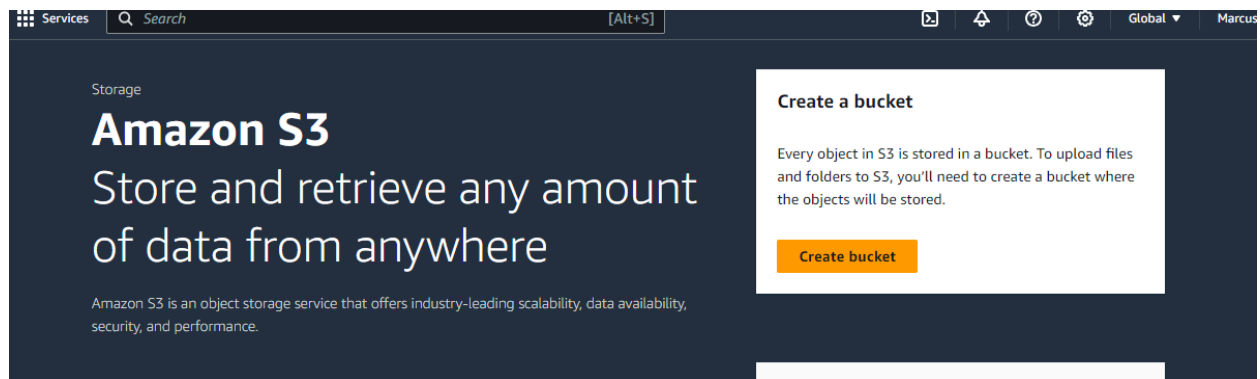
**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
  
Maximum 64 characters. Use alphanumeric and '+=,.,@-\_' characters.

**Description**  
Add a short explanation for this role.  
  
Maximum 1000 characters. Use alphanumeric and '+=,.,@-\_' characters.

### 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

## Create bucket [Info](#)

Buckets are containers for data stored in S3.

### General configuration

**AWS Region**

US East (N. Virginia) us-east-1 ▼

**Bucket type** [Info](#)

☒ **General purpose**

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory - New**

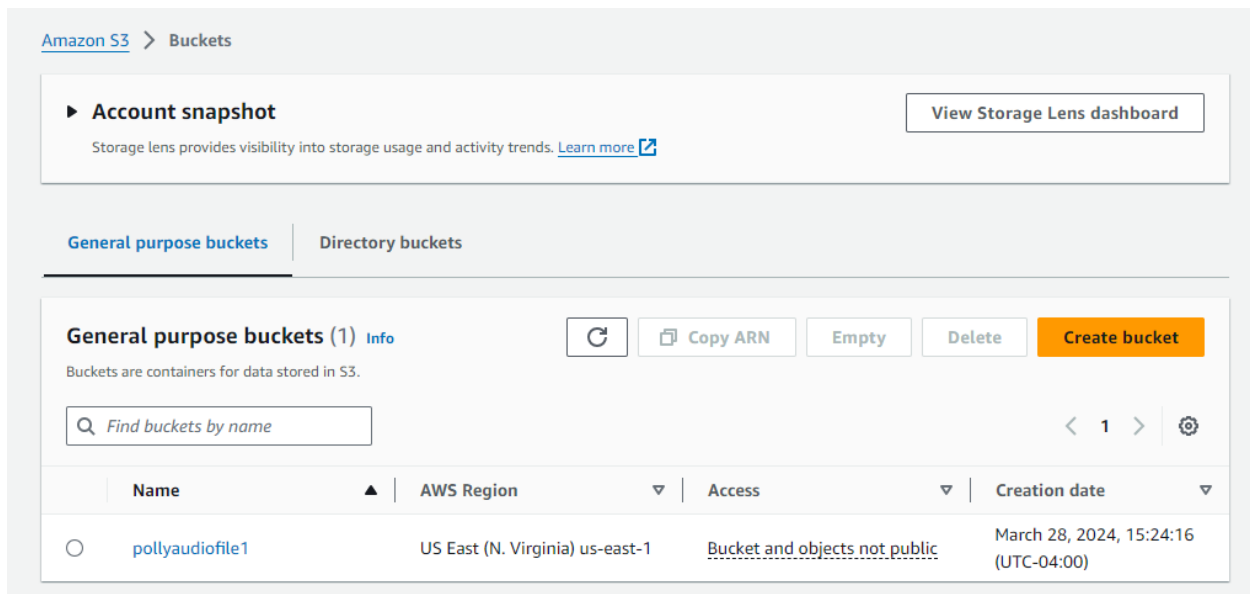
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

**Bucket name** [Info](#)

myawsbucket

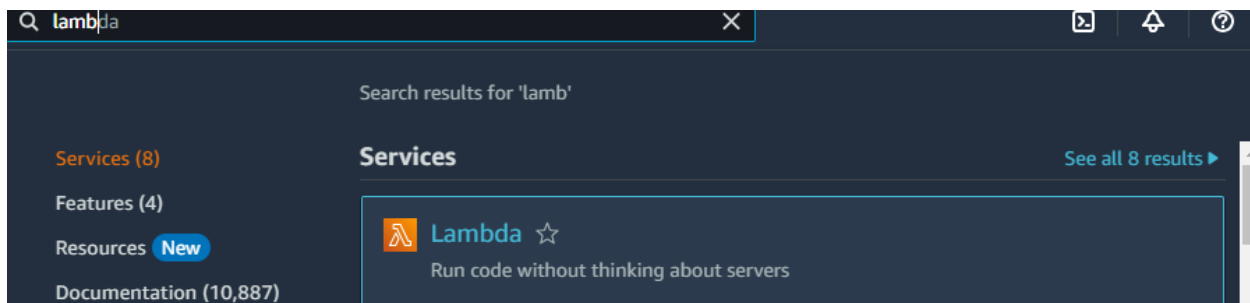
Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) [↗](#)

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.

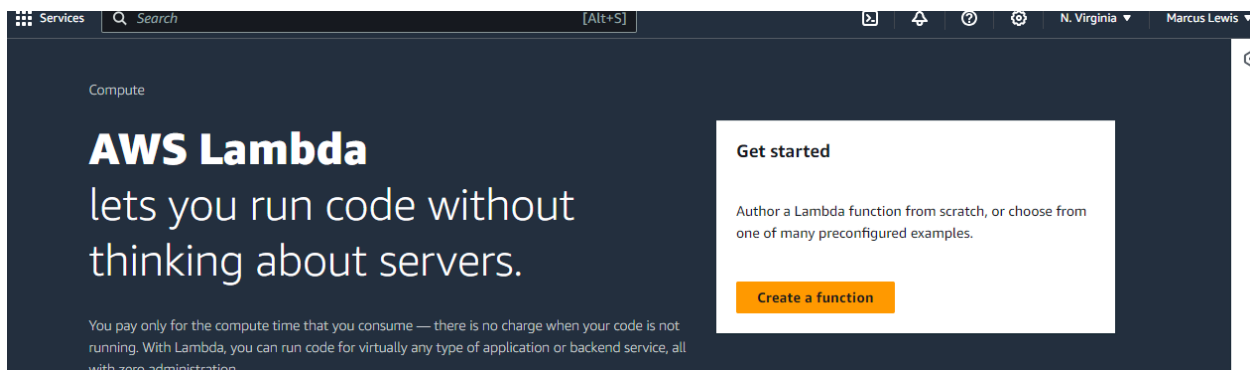


## Step 4

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

Function name  
Enter a name that describes the purpose of your function.

LambdaPolly

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 16.x

Architecture [Info](#)  
Choose the instruction set architecture you want for your function code.

☒ x86\_64  
☐ arm64

Permissions [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

---

▼ Change default execution role

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions  
☒ Use an existing role  
☐ Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

PollyTranslationRole

[View the PollyTranslationRole role](#) on the IAM console.

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 sound 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

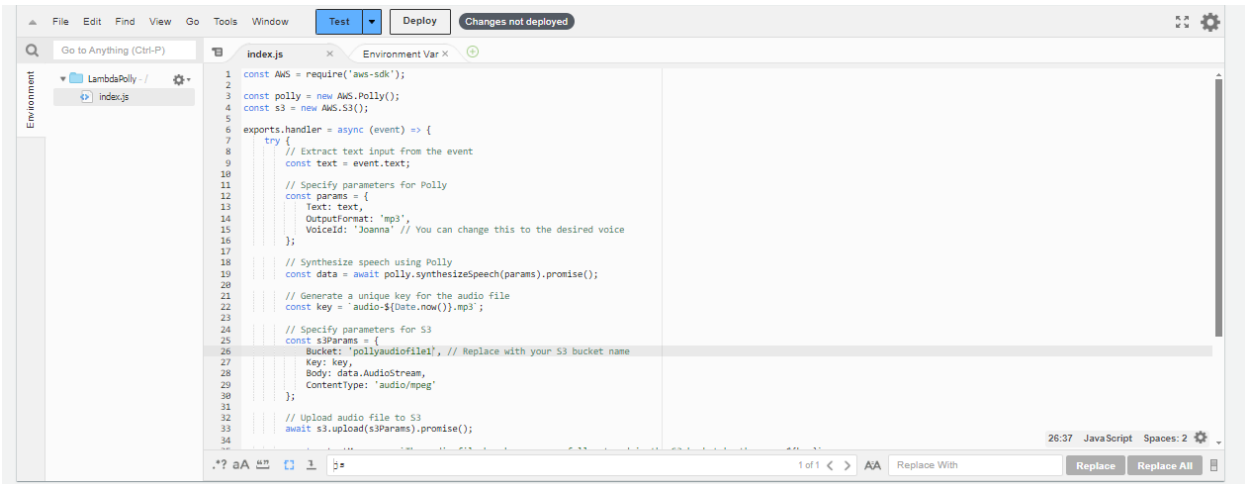
```
const outputMessage = `The audio file has been successfully stored in the S3 bucket by the name ${key}`;

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.

### Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

Event name

Pollytest1

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

#### Event JSON

Format JSON

```
1 {
2   "text": "You're this close to finishing this project! Keep it up!"
3 }
```

Cancel

Invoke

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

The test event Pollytest1 was successfully saved.

File Edit Find View Go Tools Window

Test Deploy

Go to Anything (Ctrl-P)

index.js x Environment Var x Execution result

Execution results

Test Event Name Pollytest1

Status: Succeeded | Max memory used: 87 MB | Time: 971.47 ms

Response

```
{
  "statusCode": 200,
  "body": "[\"message\": \"The audio file has been successfully stored in the S3 bucket by the name audio-1711686480966.mp3\"]"
}
```

Function Logs

```
START RequestId: 7de294d4-afe5-4b7c-a15f-7d454b18942f Version: $LATEST
END RequestId: 7de294d4-afe5-4b7c-a15f-7d454b18942f
REPORT RequestId: 7de294d4-afe5-4b7c-a15f-7d454b18942f Duration: 971.47 ms Billed Duration: 972 ms Memory Size: 128 MB Max Memory Used: 87 MB Init Duration: 484.80 ms
```

Request ID 7de294d4-afe5-4b7c-a15f-7d454b18942f

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

pollyaudiofile1 [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

---


**Objects (1)** [Info](#)

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) ▼

[Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

< 1 > ⚙️

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 <a href="#">audio-1711686404966.mp3</a>	mp3	March 29, 2024, 00:26:46 (UTC-04:00)	18.3 KB	Standard

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 Lambda from the Management Console and deleting it.

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