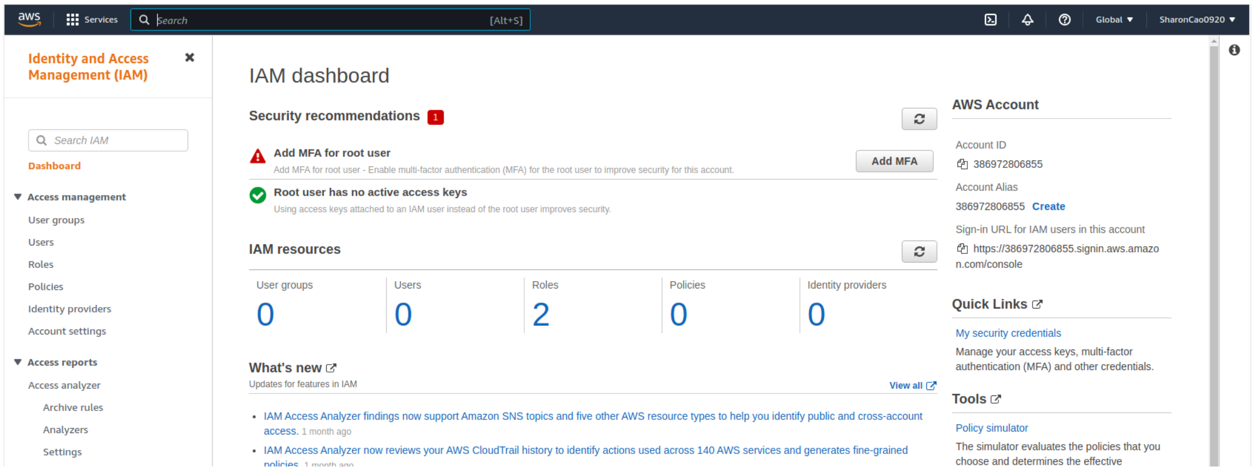
```
Q4 ==> Project: Chapter 7: Using Amazon Polly to make your sensor spea
k
```

1. Access the AWS IAM dashboard on http://console.aws.amazon.com/iam/. Then, create a new user if you don't have it yet.



aws Services Search [Alt+S] Global ▾ SharonCao0920 ▾

Identity and Access Management (IAM)

Search IAM
Dashboard
▾ Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
▾ Access reports
Access analyzer
Archive rules
Analyzers
Settings

IAM dashboard

Security recommendations 1

Add MFA for root user
Add MFA for root user - Enable multi-factor authentication (MFA) for the root user to improve security for this account.
Add MFA

Root user has no active access keys
Using access keys attached to an IAM user instead of the root user improves security.

IAM resources

User groups	Users	Roles	Policies	Identity providers
0	0	2	0	0

What's new
Updates for features in IAM
View all

- IAM Access Analyzer findings now support Amazon SNS topics and five other AWS resource types to help you identify public and cross-account access. 1 month ago
- IAM Access Analyzer now reviews your AWS CloudTrail history to identify actions used across 140 AWS services and generates fine-grained policies. 1 month ago

AWS Account

Account ID
386972806855
Account Alias
386972806855 Create
Sign-in URL for IAM users in this account
https://386972806855.signin.aws.amazon.com/console

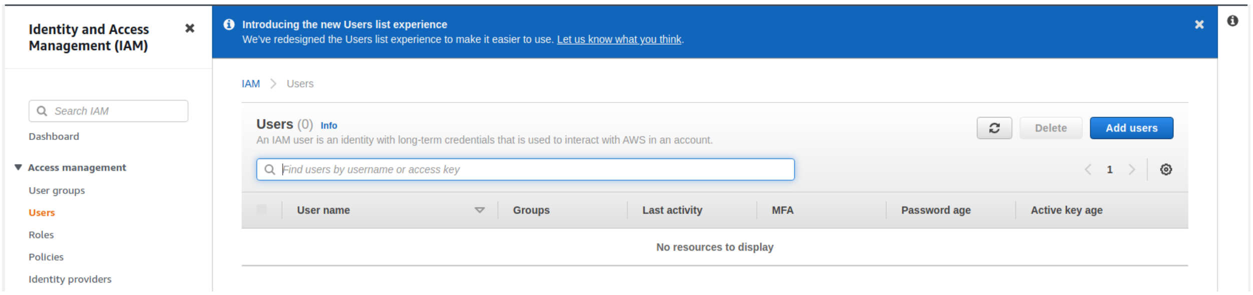
Quick Links

My security credentials
Manage your access keys, multi-factor authentication (MFA) and other credentials.

Tools

Policy simulator
The simulator evaluates the policies that you choose and determines the effective

➔ I don't have an user listed, click Add users



Identity and Access Management (IAM)

Search IAM
Dashboard
▾ Access management
User groups
Users
Roles
Policies
Identity providers

Introducing the new Users list experience
We've redesigned the Users list experience to make it easier to use. Let us know what you think.

IAM > Users

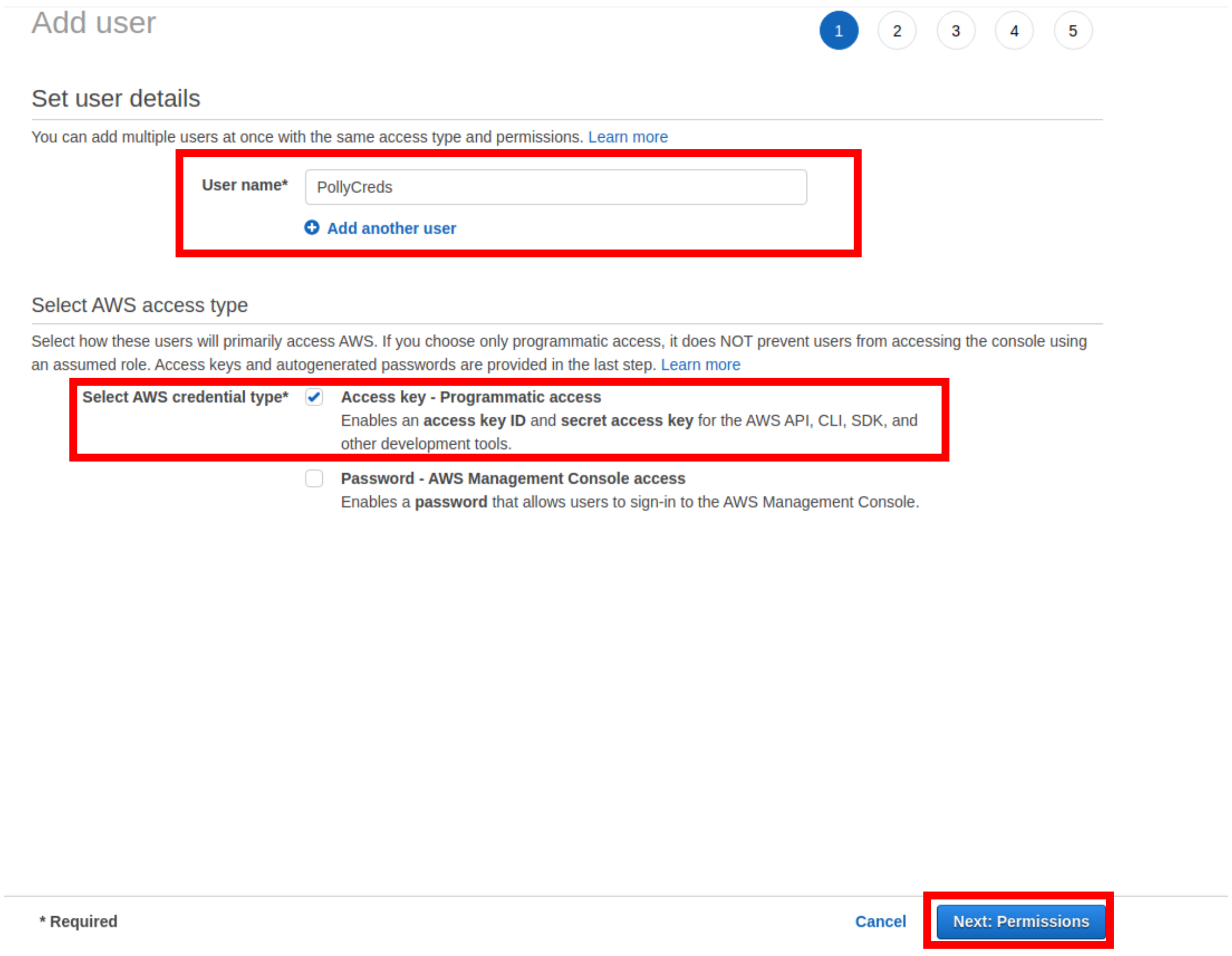
Users (0) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.
Delete Add users

Find users by username or access key

User name	Groups	Last activity	MFA	Password age	Active key age

No resources to display

➔ Follow the instructions to add user


Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* PollyCreds

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

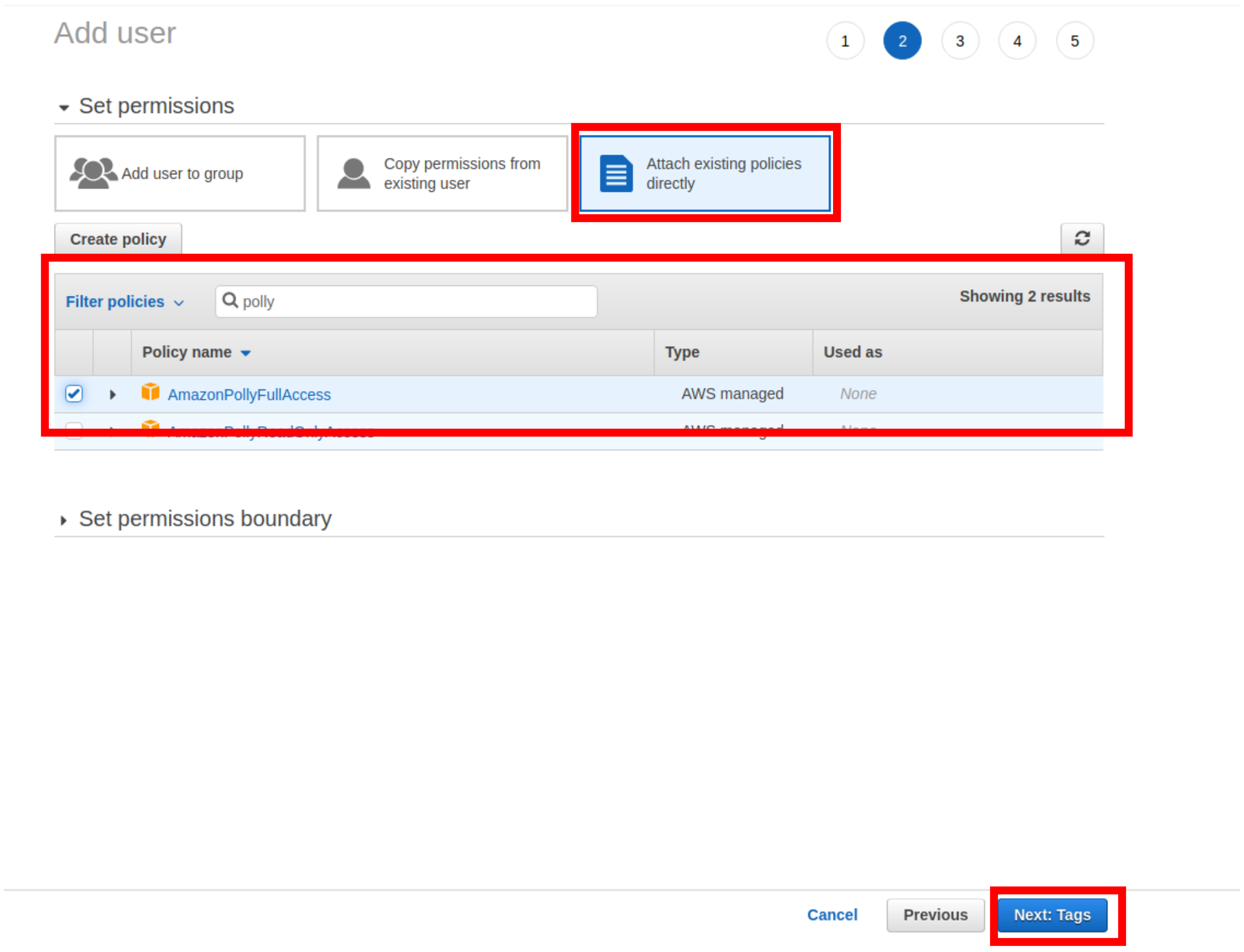
Select AWS credential type* [✓] **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

[ ] **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

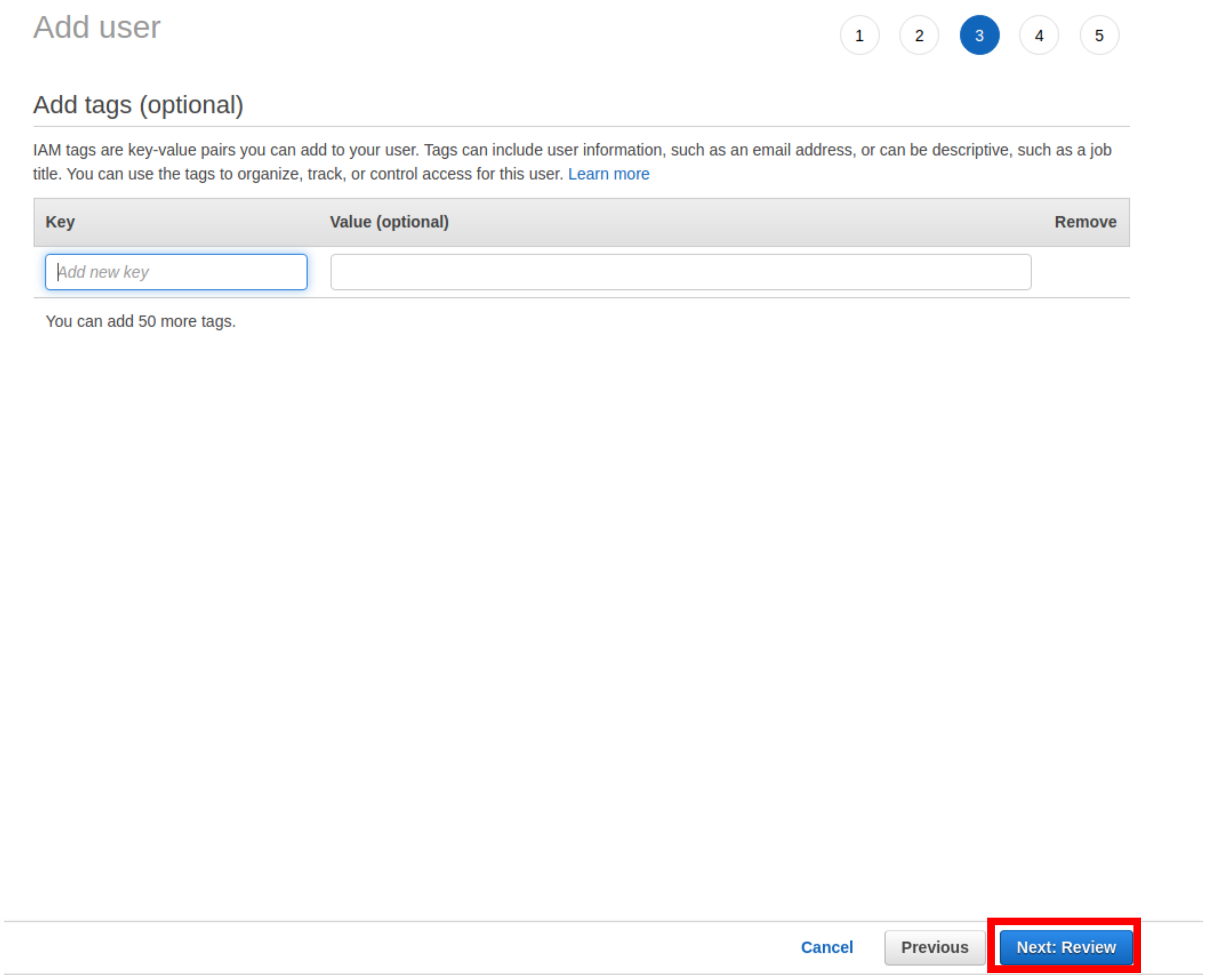
* Required

2. Now you can configure your user to give permission to access Amazon Polly.

➔ Select attach existing policies and type "polly" in search area, select and attach



		Policy name	Type	Used as
✓	▸	AmazonPollyFullAccess	AWS managed	None
	▸	AmazonPollyReadOnlyAccess	AWS managed	None

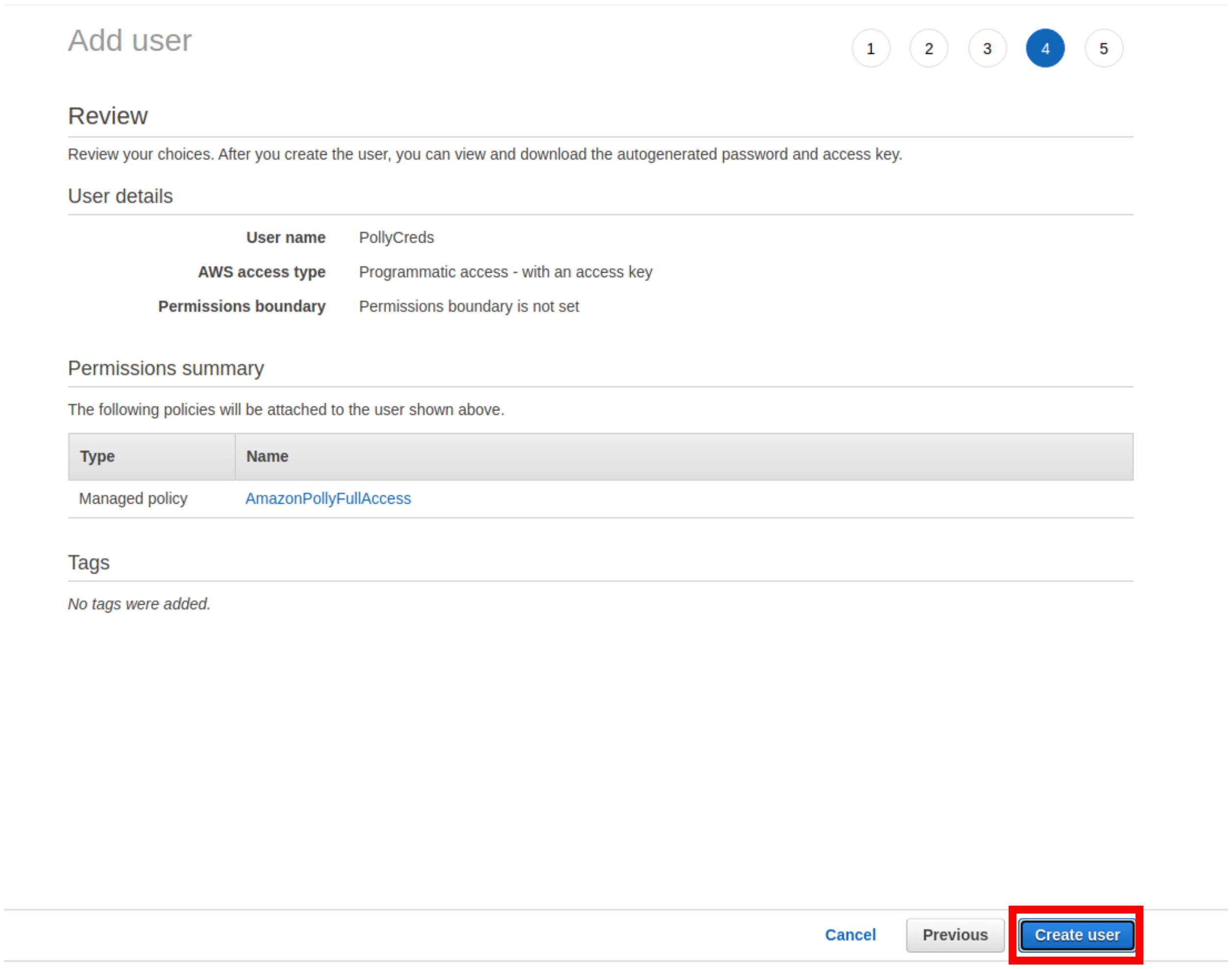


Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
Add new key		

You can add 50 more tags.



Add user

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	PollyCreds
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

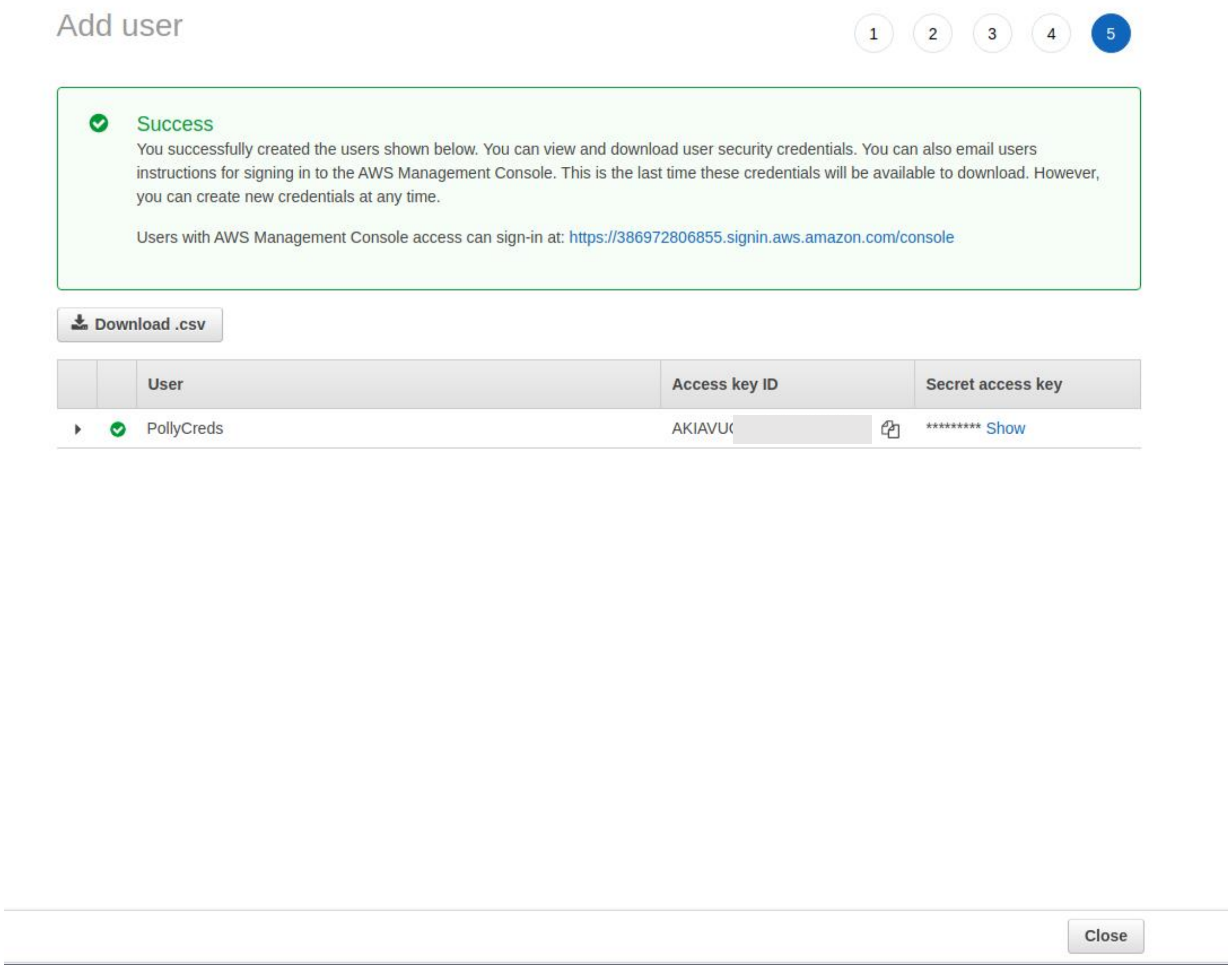
The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonPollyFullAccess

Tags

No tags were added.

➔ User created successfully, download the .csv file for records


Add user

1 2 3 4 5

Success

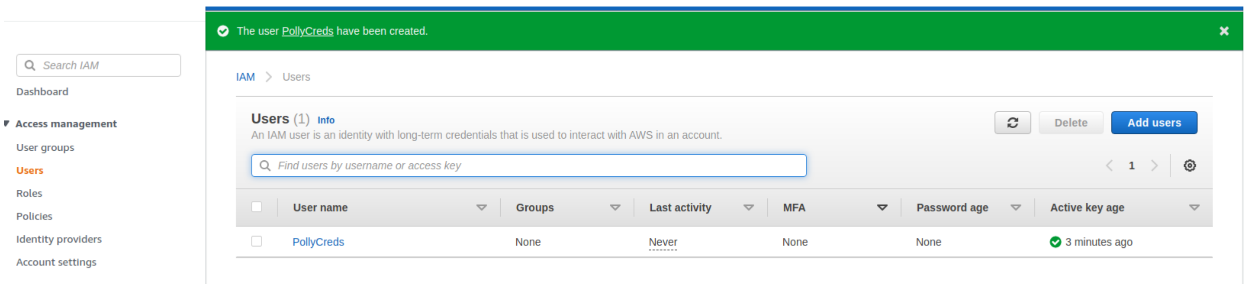
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://386972806855.signin.aws.amazon.com/console

Download .csv

		User	Access key ID	Secret access key
▸	✓	PollyCreds	AKIAVU[redacted]	********* Show

Close


The user PollyCreds have been created.

Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings

IAM > Users

Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete Add users

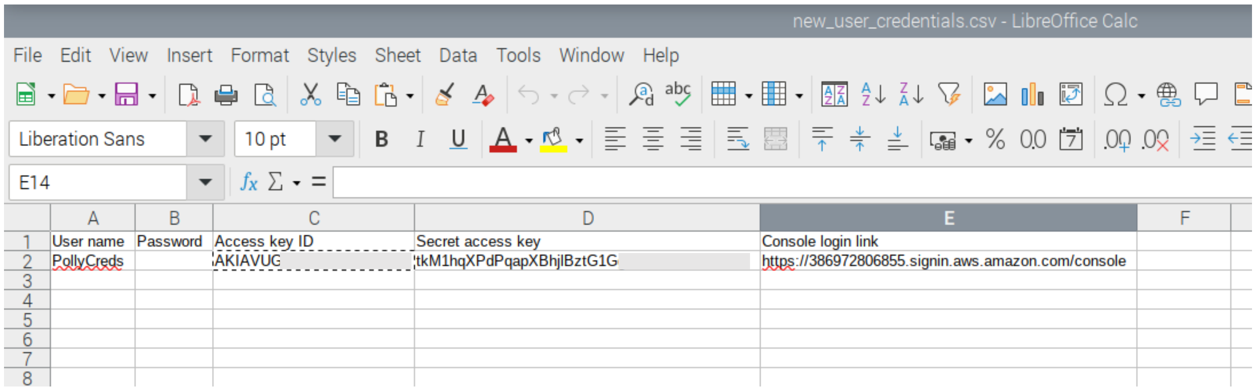
Find users by username or access key

	User name	Groups	Last activity	MFA	Password age	Active key age
☐	PollyCreds	None	Never	None	None	✓ 3 minutes ago

User added successfully and shown on dashboard!

3. Next, you should copy the AWS access key ID from your IAM user. You can find it under the Security credentials tab. You can create an AWS access key if you don't have it. This AWS access key ID will be used in our program.

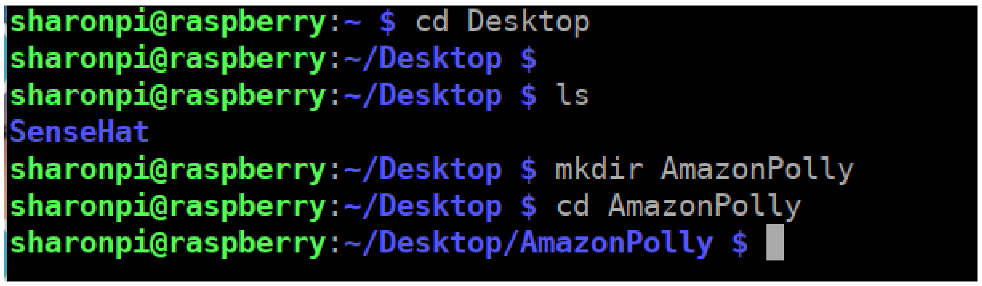
➔ I get my access keys from the previous file downloaded


new_user_credentials.csv - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
E14

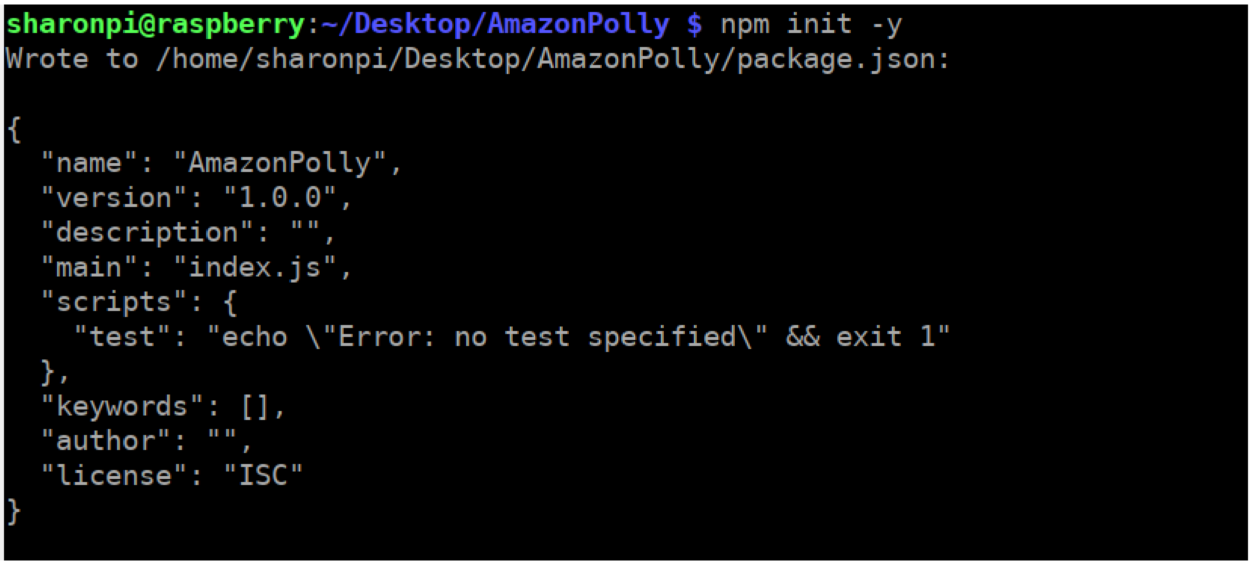
	A	B	C	D	E	F
1	User name	Password	Access key ID	Secret access key	Console login link	
2	PollyCreds		AKIAVUG[redacted]	tkM1hqXPdPqapXBhjlBztG1G[redacted]	https://386972806855.signin.aws.amazon.com/console	
3						
4						
5						
6						
7						
8						

4. For testing, we use Node.js to develop a program. We need AWS SDK for JavaScript/Node.js to access Amazon Polly.

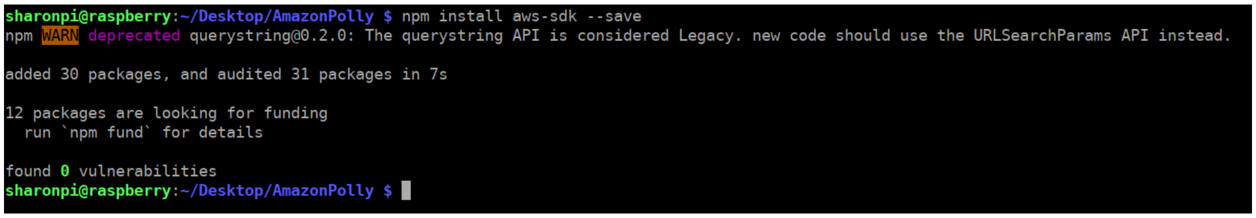
```
$ mkdir ml
$ cd ml/
$ npm init (npm init -y to use default options)
$ npm install aws-sdk -save
```



```
sharonpi@raspberry:~ $ cd Desktop
sharonpi@raspberry:~/Desktop $
sharonpi@raspberry:~/Desktop $ ls
SenseHat
sharonpi@raspberry:~/Desktop $ mkdir AmazonPolly
sharonpi@raspberry:~/Desktop $ cd AmazonPolly
sharonpi@raspberry:~/Desktop/AmazonPolly $
```



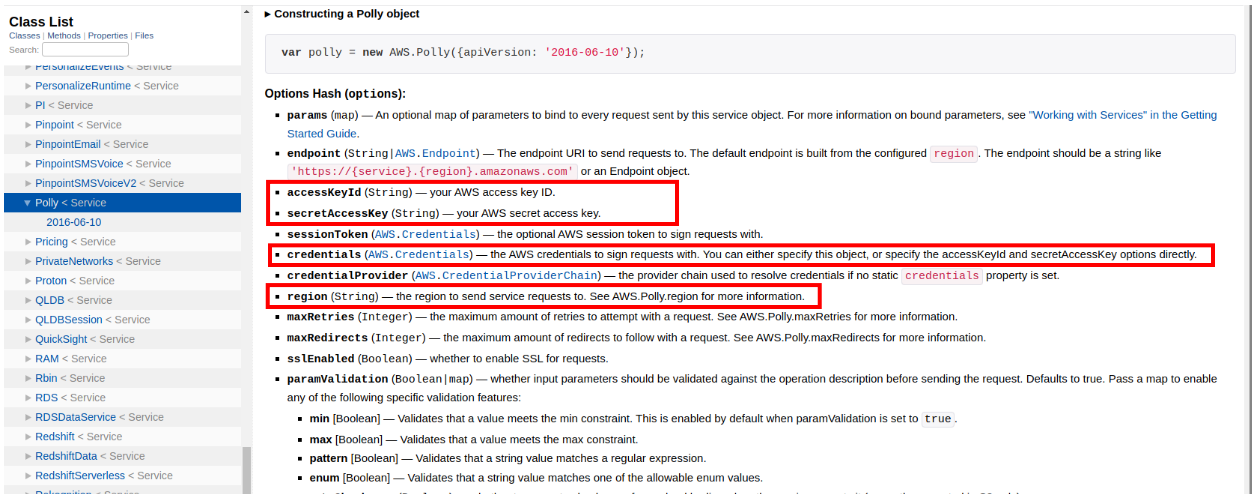
```
sharonpi@raspberry:~/Desktop/AmazonPolly $ npm init -y
Wrote to /home/sharonpi/Desktop/AmazonPolly/package.json:

{
  "name": "AmazonPolly",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```



```
sharonpi@raspberry:~/Desktop/AmazonPolly $ npm install aws-sdk --save
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

added 30 packages, and audited 31 packages in 7s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
sharonpi@raspberry:~/Desktop/AmazonPolly $ 
```

Project folder ready!

5. We will use the `Polly` object to access AWS Polly from Node.js. You can read more information about the `Polly` object on https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Polly.html. We pass our AWS access key ID to perform AWS authentication.



▸ Constructing a Polly object

```
var polly = new AWS.Polly({apiVersion: '2016-06-10'});
```

Options Hash (options):

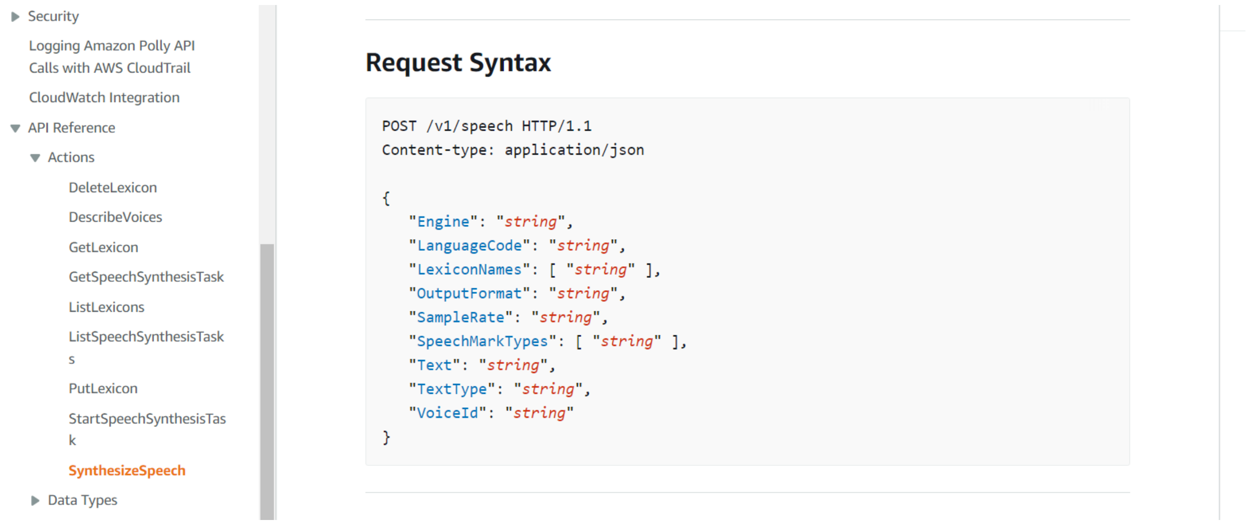
- **params** (map) — An optional map of parameters to bind to every request sent by this service object. For more information on bound parameters, see "Working with Services" in the Getting Started Guide.
- **endpoint** (String | AWS.Endpoint) — The endpoint URI to send requests to. The default endpoint is built from the configured `region`. The endpoint should be a string like `'https://{service}.{region}.amazonaws.com'` or an Endpoint object.
- **accessKeyId** (String) — your AWS access key ID.
- **secretAccessKey** (String) — your AWS secret access key.
- **sessionToken** (AWS.Credentials) — the optional AWS session token to sign requests with.
- **credentials** (AWS.Credentials) — the AWS credentials to sign requests with. You can either specify this object, or specify the accessKeyId and secretAccessKey options directly.
- **credentialProvider** (AWS.CredentialProviderChain) — the provider chain used to resolve credentials if no static `credentials` property is set.
- **region** (String) — the region to send service requests to. See AWS.Polly.region for more information.
- **maxRetries** (Integer) — the maximum amount of retries to attempt with a request. See AWS.Polly.maxRetries for more information.
- **maxRedirects** (Integer) — the maximum amount of redirects to follow with a request. See AWS.Polly.maxRedirects for more information.
- **sslEnabled** (Boolean) — whether to enable SSL for requests.
- **paramValidation** (Boolean | map) — whether input parameters should be validated against the operation description before sending the request. Defaults to true. Pass a map to enable any of the following specific validation features:
  - **min** [Boolean] — Validates that a value meets the min constraint. This is enabled by default when paramValidation is set to `true`.
  - **max** [Boolean] — Validates that a value meets the max constraint.
  - **pattern** [Boolean] — Validates that a string value matches a regular expression.
  - **enum** [Boolean] — Validates that a string value matches one of the allowable enum values.

➔ You can use credentials to give the sign request, but I will give the keys directly.

6. To convert from text-to-speech, we can call `Polly.synthesizeSpeech()`. From this process, we can save the result into an MP3 file.

➔ Go to Amazon Polly Documentation to learn more about the Request body structure.

https://docs.aws.amazon.com/polly/latest/dg/API_SynthesizeSpeech.html



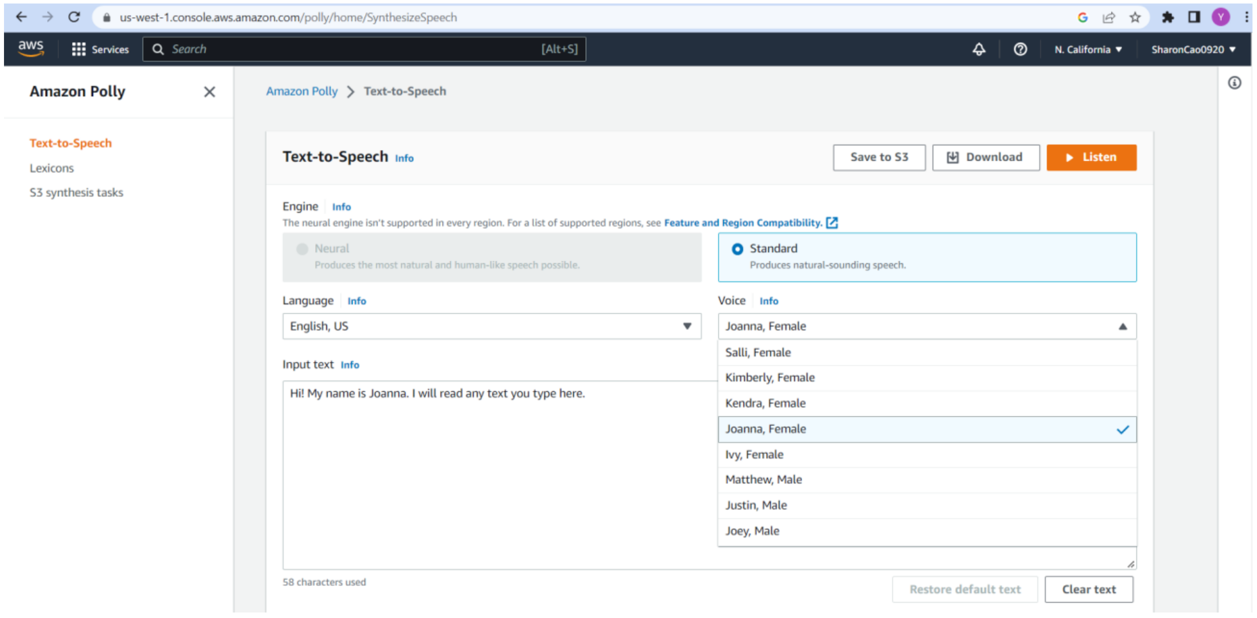
Request Syntax

```
POST /v1/speech HTTP/1.1
Content-type: application/json

{
   "Engine": "string",
   "LanguageCode": "string",
   "LexiconNames": [ "string" ],
   "OutputFormat": "string",
   "SampleRate": "string",
   "SpeechMarkTypes": [ "string" ],
   "Text": "string",
   "TextType": "string",
   "VoiceId": "string"
}
```

➔ You can try and choose the voice you like for speech.

Go to aws console -> Services -> All Services -> Amazon Polly -> Try Polly

Select and try your desired VoiceId, languages and so on



Text-to-Speech Info

Engine Info
The neural engine isn't supported in every region. For a list of supported regions, see Feature and Region Compatibility.

Neural — Produces the most natural and human-like speech possible.
[✓ Standard — Produces natural-sounding speech.]

Language Info: English, US

Voice Info: Joanna, Female

- Salli, Female
- Kimberly, Female
- Kendra, Female
- Joanna, Female ✓
- Ivy, Female
- Matthew, Male
- Justin, Male
- Joey, Male

Input text Info

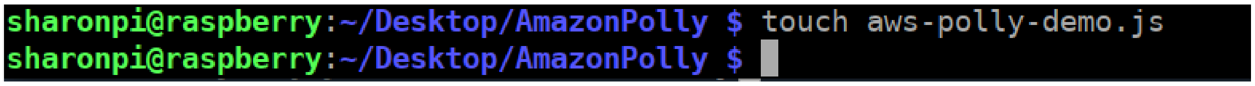
Hi! My name is Joanna. I will read any text you type here.

58 characters used

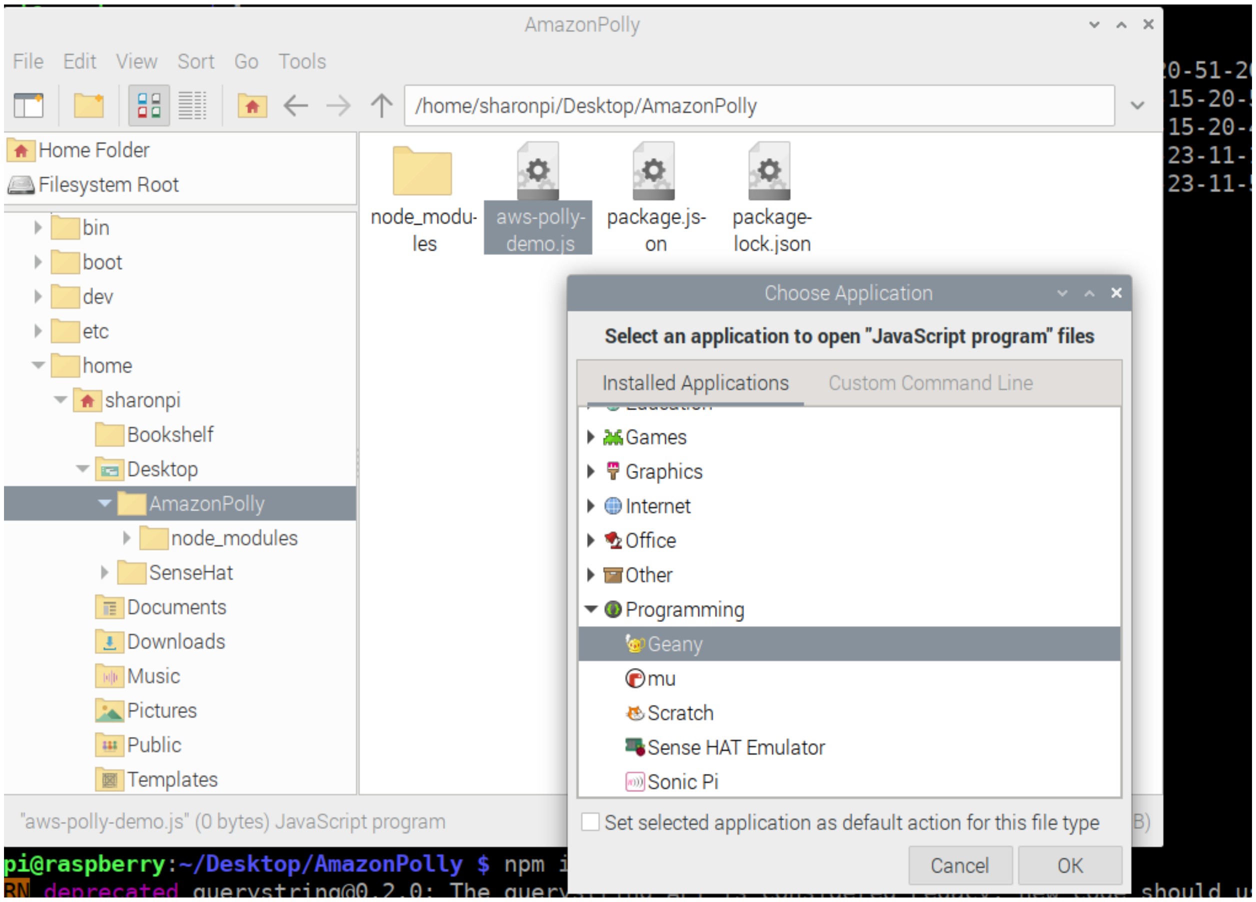
Restore default text | Clear text

7. Let's create a file, `aws-polly-demo.js`.

➔ Create and edit aws-polly-demo.js



```
sharonpi@raspberry:~/Desktop/AmazonPolly $ touch aws-polly-demo.js
sharonpi@raspberry:~/Desktop/AmazonPolly $ 
```



```
/home/sharonpi/Desktop/AmazonPolly
```

Select an application to open "JavaScript program" files

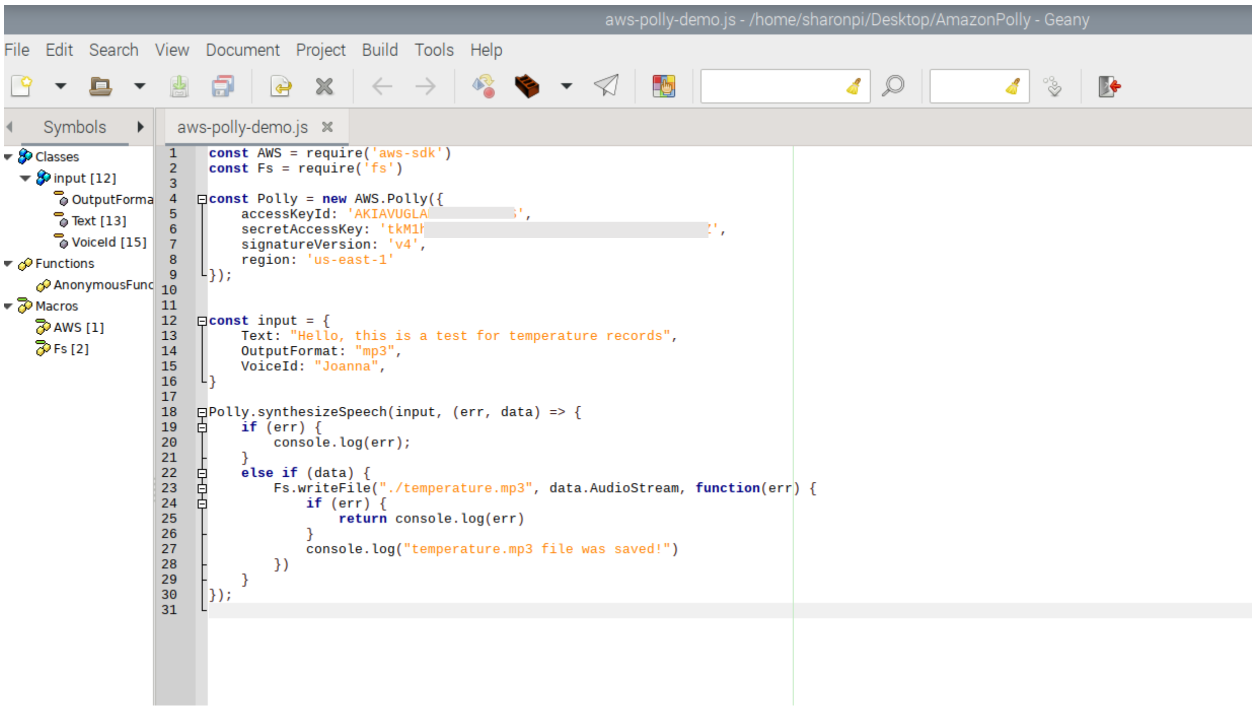
☐ Set selected application as default action for this file type

"aws-polly-demo.js" (0 bytes) JavaScript program

```
pi@raspberry:~/Desktop/AmazonPolly $ npm i
RN deprecated querystring@0.2.0: The query
```

➔ Change the credentials and Input values

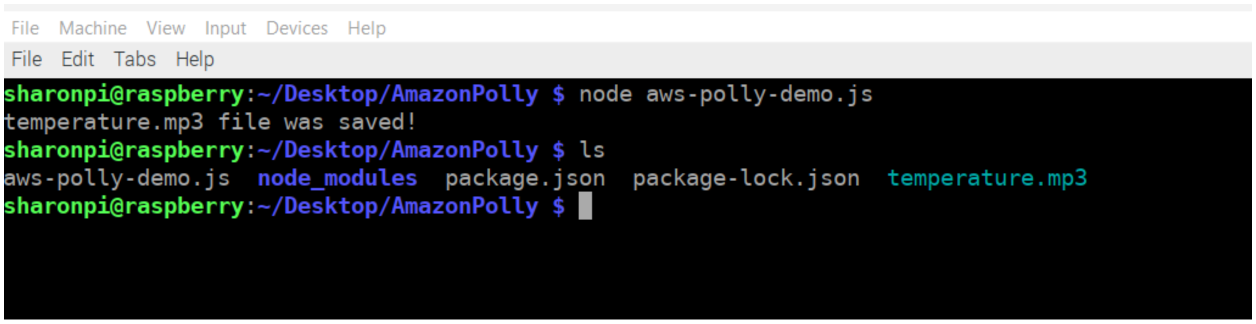
```
const Polly = new AWS.Polly({
accessKeyId: 'xxxxx',
secretAccessKey: xxxxx',
signatureVersion: 'v4',
region: 'us-east-1'
});


const input = {
Text: "Hello, this is a test for temperature records",
OutputFormat: "mp3",
VoiceId: "Joanna",
}
```


aws-polly-demo.js - /home/sharonpi/Desktop/AmazonPolly - Geany
File Edit Search View Document Project Build Tools Help
Symbols
aws-polly-demo.js
Classes
input [12]
OutputForma
Text [13]
VoiceId [15]
Functions
AnonymousFunc
Macros
AWS [1]
Fs [2]

```
const AWS = require('aws-sdk')
const Fs = require('fs')

const Polly = new AWS.Polly({
    accessKeyId: 'AKIAVUGLA[redacted]',
    secretAccessKey: 'tkM1[redacted]',
    signatureVersion: 'v4',
    region: 'us-east-1'
});


const input = {
    Text: "Hello, this is a test for temperature records",
    OutputFormat: "mp3",
    VoiceId: "Joanna",
}

Polly.synthesizeSpeech(input, (err, data) => {
    if (err) {
        console.log(err);
    }
    else if (data) {
        Fs.writeFile("./temperature.mp3", data.AudioStream, function(err) {
            if (err) {
                return console.log(err)
            }
            console.log("temperature.mp3 file was saved!")
        })
    }
});
```

8. Save this program and run it using the following command:

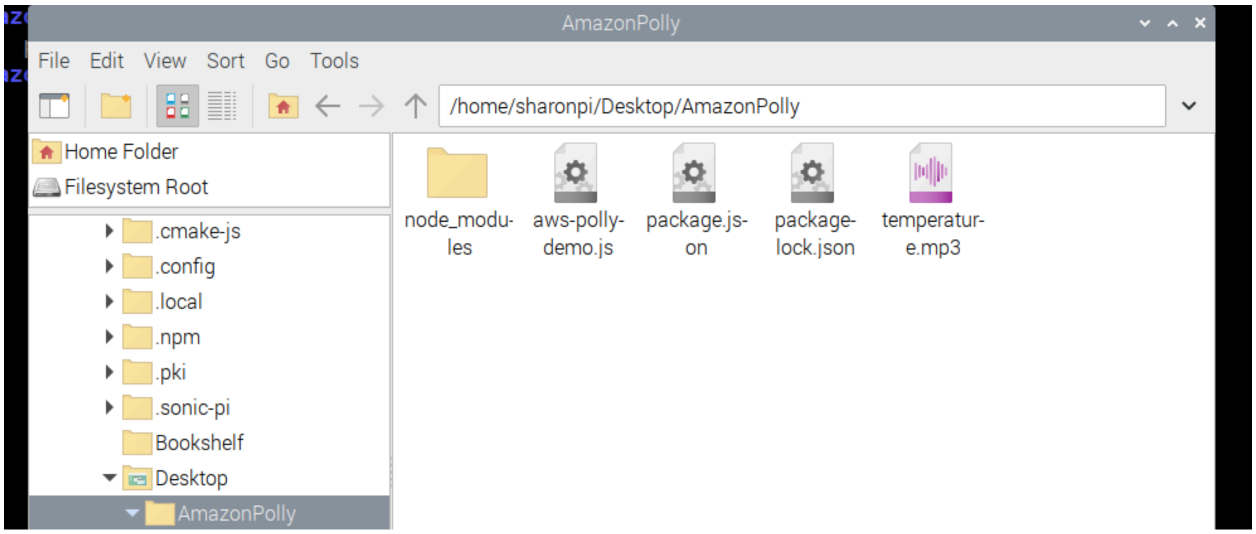
```
$ node aws-polly-demo.js
```

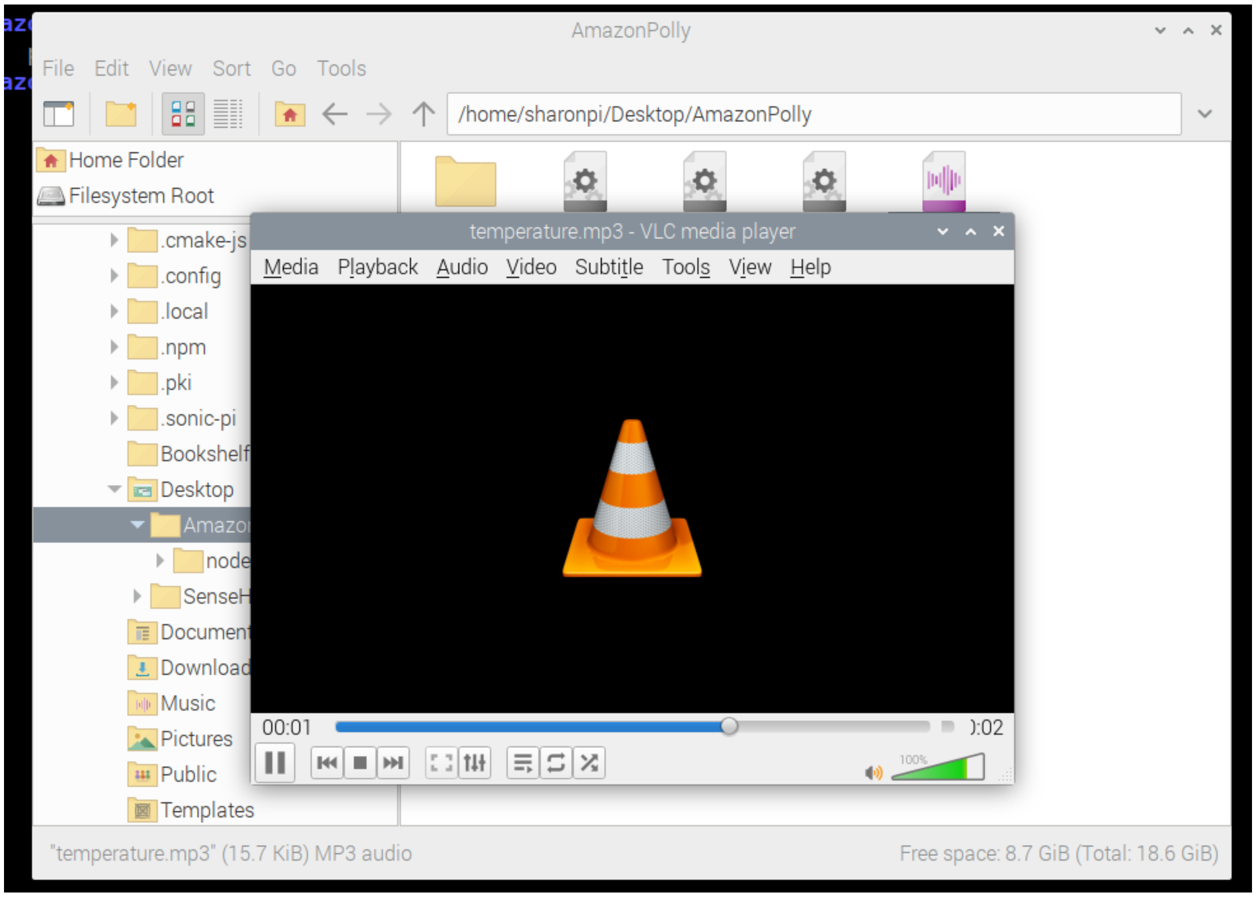


```
sharonpi@raspberry:~/Desktop/AmazonPolly $ node aws-polly-demo.js
temperature.mp3 file was saved!
sharonpi@raspberry:~/Desktop/AmazonPolly $ ls
aws-polly-demo.js  node_modules  package.json  package-lock.json  temperature.mp3
sharonpi@raspberry:~/Desktop/AmazonPolly $ 
```

Mp3 file is created successfully!

9. You should see the MP3 file from the executing result. You can see the program output that is shown in the following screenshot. Try to run that MP3 file:





"temperature.mp3" (15.7 KiB) MP3 audio

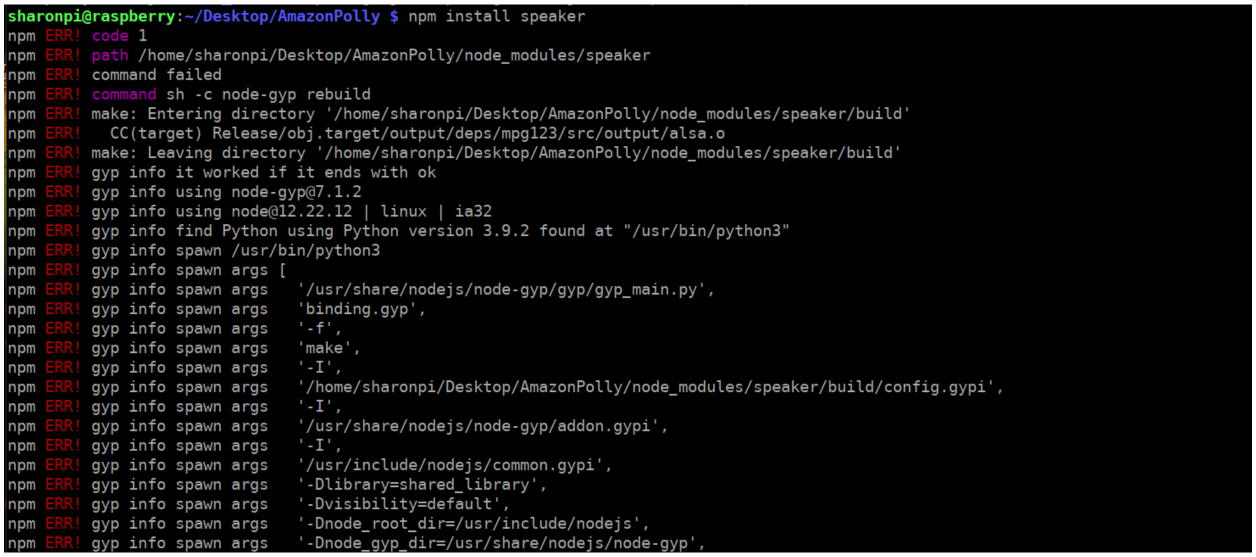
Free space: 8.7 GiB (Total: 18.6 GiB)

Mp3 file reading the input file is playing with no issue!

10. Use node-speaker library

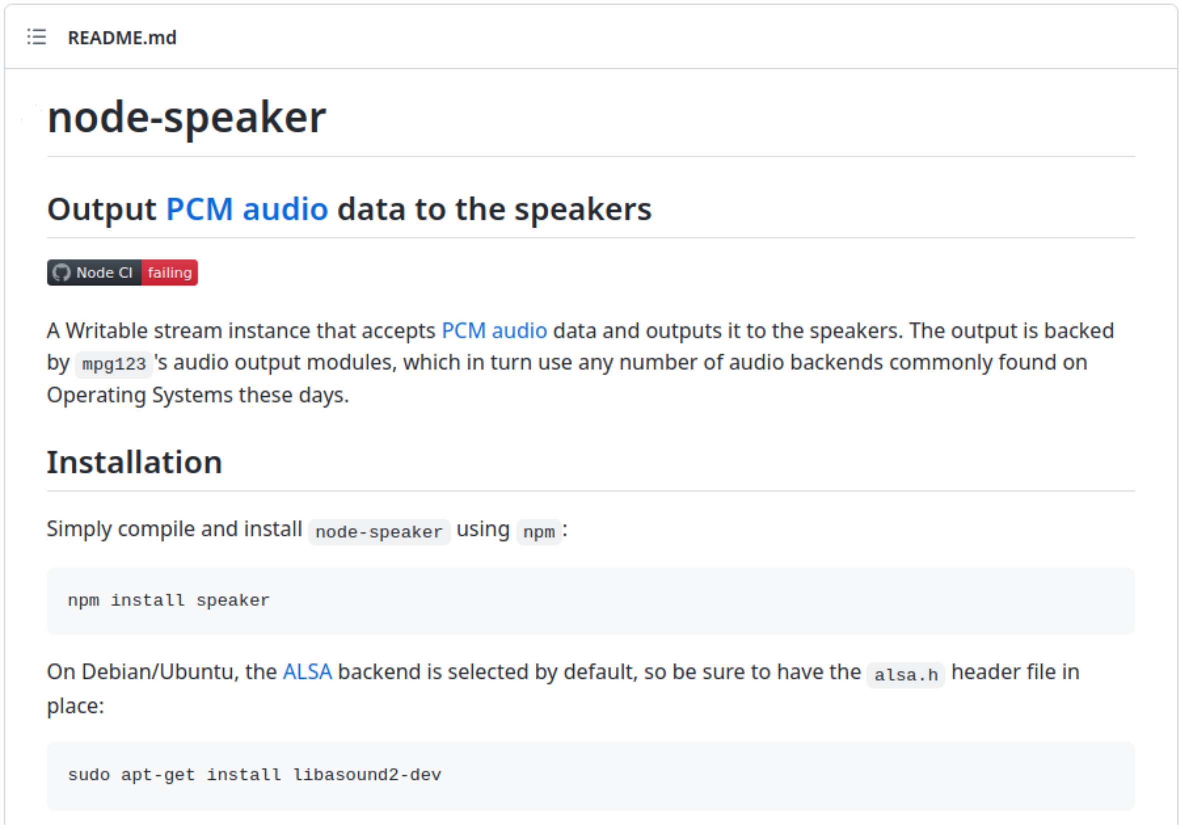
➔ Install node-speaker library with npm

```
$ npm install speaker
```



```
sharonpi@raspberry:~/Desktop/AmazonPolly $ npm install speaker
npm ERR! code 1
npm ERR! path /home/sharonpi/Desktop/AmazonPolly/node_modules/speaker
npm ERR! command failed
npm ERR! command sh -c node-gyp rebuild
npm ERR! make: Entering directory '/home/sharonpi/Desktop/AmazonPolly/node_modules/speaker/build'
npm ERR!   CC(target) Release/obj.target/output/deps/mpg123/src/output/alsa.o
npm ERR! make: Leaving directory '/home/sharonpi/Desktop/AmazonPolly/node_modules/speaker/build'
npm ERR! gyp info it worked if it ends with ok
npm ERR! gyp info using node-gyp@7.1.2
npm ERR! gyp info using node@12.22.12 | linux | ia32
npm ERR! gyp info find Python using Python version 3.9.2 found at "/usr/bin/python3"
npm ERR! gyp info spawn /usr/bin/python3
npm ERR! gyp info spawn args [
npm ERR! gyp info spawn args   '/usr/share/nodejs/node-gyp/gyp/gyp_main.py',
npm ERR! gyp info spawn args   'binding.gyp',
npm ERR! gyp info spawn args   '-f',
npm ERR! gyp info spawn args   'make',
npm ERR! gyp info spawn args   '-I',
npm ERR! gyp info spawn args   '/home/sharonpi/Desktop/AmazonPolly/node_modules/speaker/build/config.gypi',
npm ERR! gyp info spawn args   '-I',
npm ERR! gyp info spawn args   '/usr/share/nodejs/node-gyp/addon.gypi',
npm ERR! gyp info spawn args   '-I',
npm ERR! gyp info spawn args   '/usr/include/nodejs/common.gypi',
npm ERR! gyp info spawn args   '-Dlibrary=shared_library',
npm ERR! gyp info spawn args   '-Dvisibility=default',
npm ERR! gyp info spawn args   '-Dnode_root_dir=/usr/include/nodejs',
npm ERR! gyp info spawn args   '-Dnode_gyp_dir=/usr/share/nodejs/node-gyp',
```

Got error installing package – info from https://github.com/TooTallNate/node-speaker



README.md

node-speaker

Output PCM audio data to the speakers

A Writable stream instance that accepts PCM audio data and outputs it to the speakers. The output is backed by `mpg123`'s audio output modules, which in turn use any number of audio backends commonly found on Operating Systems these days.

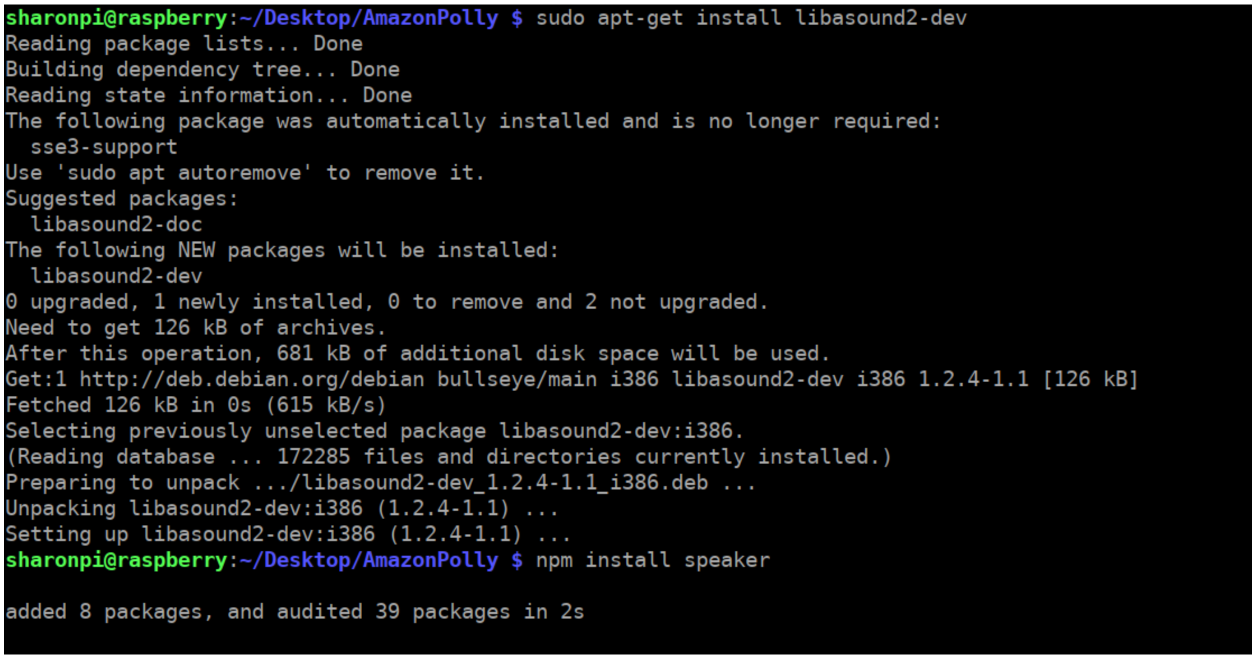
Installation

Simply compile and install `node-speaker` using `npm`:

```
npm install speaker
```

On Debian/Ubuntu, the ALSA backend is selected by default, so be sure to have the `alsa.h` header file in place:

```
sudo apt-get install libasound2-dev
```



```
sharonpi@raspberry:~/Desktop/AmazonPolly $ sudo apt-get install libasound2-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  sse3-support
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  libasound2-doc
The following NEW packages will be installed:
  libasound2-dev
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 126 kB of archives.
After this operation, 681 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main i386 libasound2-dev i386 1.2.4-1.1 [126 kB]
Fetched 126 kB in 0s (615 kB/s)
Selecting previously unselected package libasound2-dev:i386.
(Reading database ... 172285 files and directories currently installed.)
Preparing to unpack .../libasound2-dev_1.2.4-1.1_i386.deb ...
Unpacking libasound2-dev:i386 (1.2.4-1.1) ...
Setting up libasound2-dev:i386 (1.2.4-1.1) ...
sharonpi@raspberry:~/Desktop/AmazonPolly $ npm install speaker

added 8 packages, and audited 39 packages in 2s
```

Package installation successfully!

11. If you are working on macOS, you will probably get errors. You can run this command to solve the error on macOS:

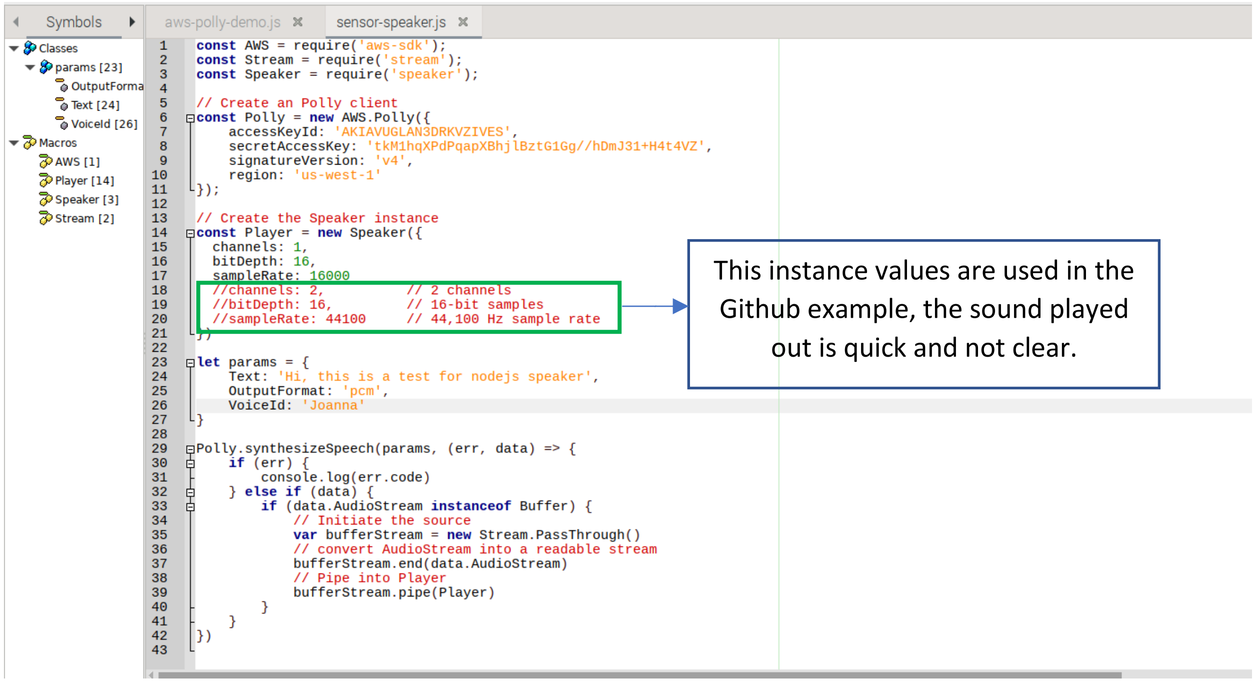
    ```
    $ XX=clang++ npm install speaker --mpg123-backend=openal
    ```

12. Now we modify our previous program to play text-to-speech streaming into node-speaker library. Create the `sensor-speaker.js` file

➔ Create and modify sensor-speaker.js file

```
// Create an Polly client
const Polly = new AWS.Polly({
accessKeyId: 'xxxxxxx',
secretAccessKey: xxxxxxx',
signatureVersion: 'v4',
region: 'us-west-1'
});

// Create the Speaker instance
const Player = new Speaker({
  channels: 1,
  bitDepth: 16,
  sampleRate: 16000
  //channels: 2,          // 2 channels
  //bitDepth: 16,         // 16-bit samples
```

```
  //sampleRate: 44100     // 44,100 Hz sample rate
})

let params = {
   Text: 'Hi, this is a test for nodejs speaker',
   OutputFormat: 'pcm',
   VoiceId: 'Joanna'
}
```

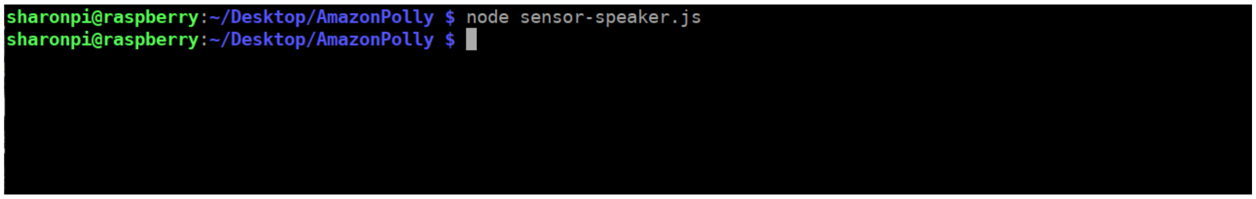


```
const AWS = require('aws-sdk');
const Stream = require('stream');
const Speaker = require('speaker');

// Create an Polly client
const Polly = new AWS.Polly({
    accessKeyId: 'AKIAVUGLAN3DRKVZIVES',
    secretAccessKey: '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',
    signatureVersion: 'v4',
    region: 'us-west-1'
});

// Create the Speaker instance
const Player = new Speaker({
  channels: 1,
  bitDepth: 16,
  sampleRate: 16000
  //channels: 2,          // 2 channels
  //bitDepth: 16,         // 16-bit samples
  //sampleRate: 44100     // 44,100 Hz sample rate
})

let params = {
    Text: 'Hi, this is a test for nodejs speaker',
    OutputFormat: 'pcm',
    VoiceId: 'Joanna'
}

Polly.synthesizeSpeech(params, (err, data) => {
    if (err) {
        console.log(err.code)
    } else if (data) {
        if (data.AudioStream instanceof Buffer) {
            // Initiate the source
            var bufferStream = new Stream.PassThrough()
            // convert AudioStream into a readable stream
            bufferStream.end(data.AudioStream)
            // Pipe into Player
            bufferStream.pipe(Player)
        }
    }
})
```

This instance values are used in the Github example, the sound played out is quick and not clear.

13. Now you can run this program by typing the following command:

```
$ node sensor-speaker.js
```



```
sharonpi@raspberry:~/Desktop/AmazonPolly $ node sensor-speaker.js
sharonpi@raspberry:~/Desktop/AmazonPolly $
```

Successfully listen the text input!