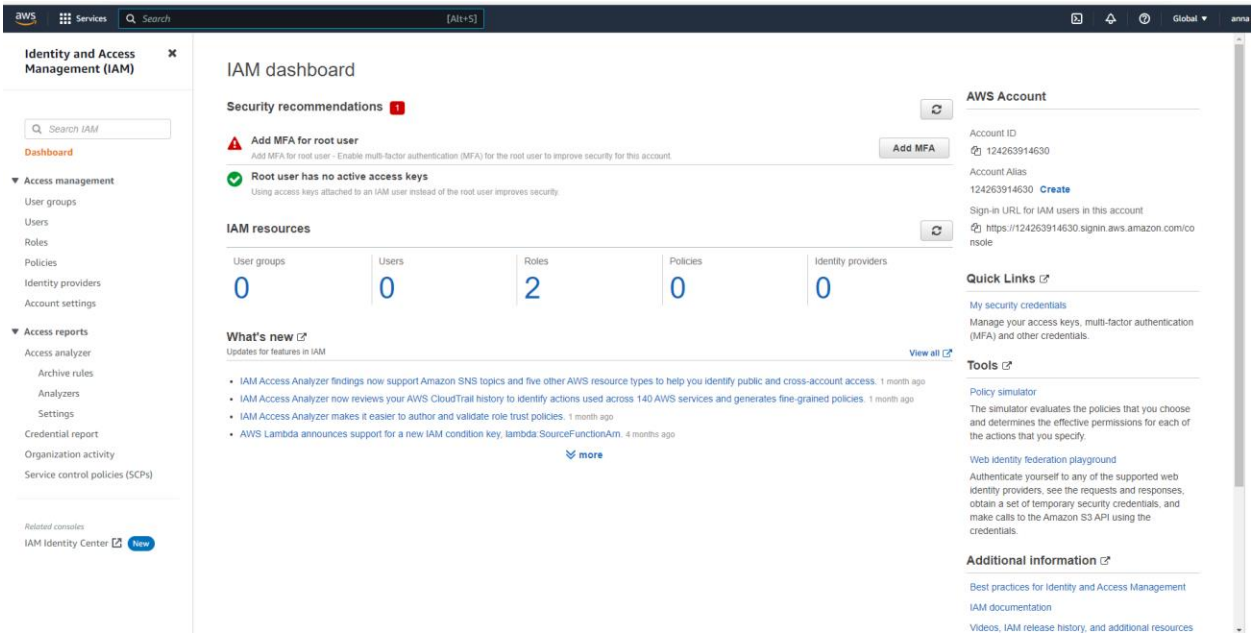
Week 12: Quiz: Chapter 7: Using Amazon Polly to make your sensor speak

4. Project: Chapter 7: Using Amazon Polly to make your sensor speak

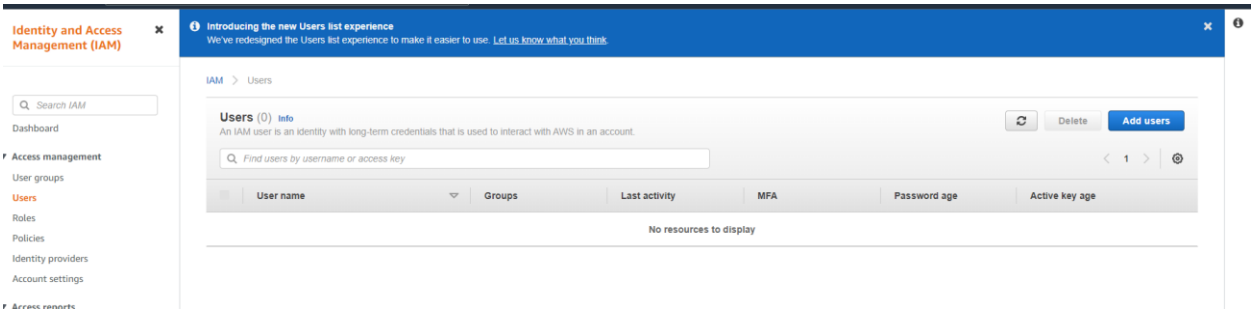
    - Step 1: Prepare Raspberry Pi emulator + VirtualBox + Sense HAT Emulator
        - References
            - DHT22 for Raspberry Pi
                - Adafruit_Python_DHT
            - Raspberry Pi Tutorial: How to Use the DHT-22
    - Step 2: Continue the proces of Making your sensor speak
    - Step 3: Update your portfolio about this project
    - Step 4: Submit a PDF file document showing the procedure as part of the homework answers.
    - Step 5: Submit the URL of your GitHub webpage as part of the homework answers.
        - GitHub directory structure
            -
            - `IoT`
            - `AWS IoT + Raspberry Pi Emulator + Text to Speech`

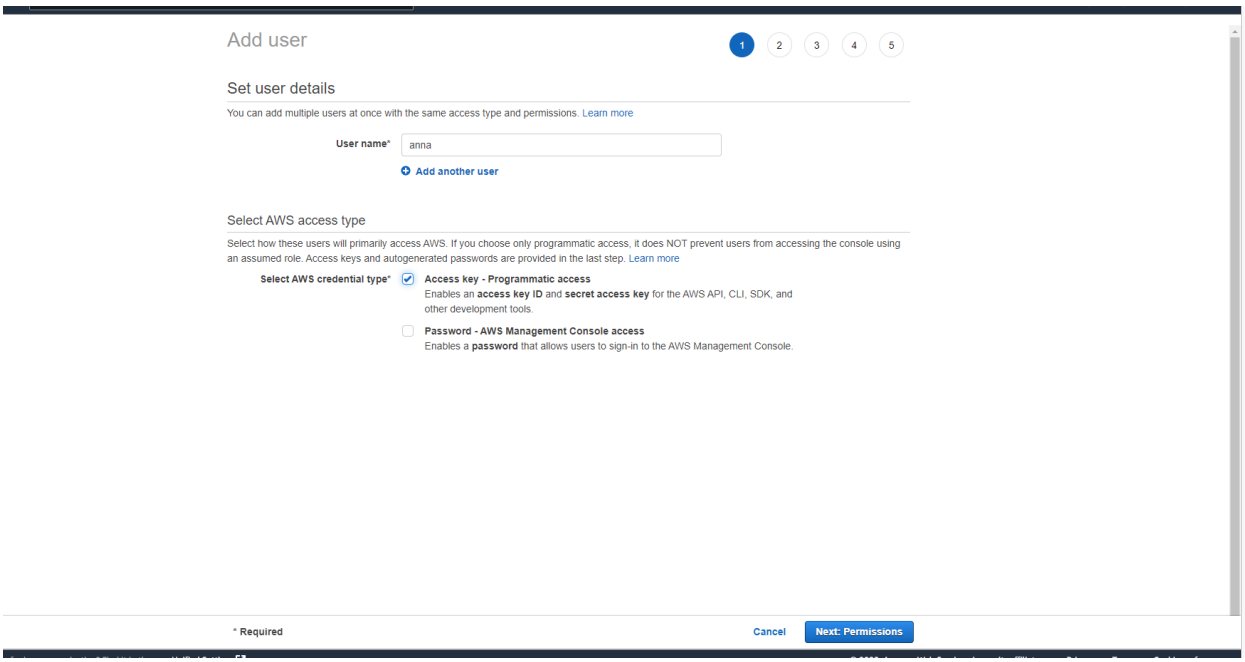
    - References
        - 2022 Fall

1. Access the AWS IAM dashboard on http://console.aws.amazon.com/iam/.



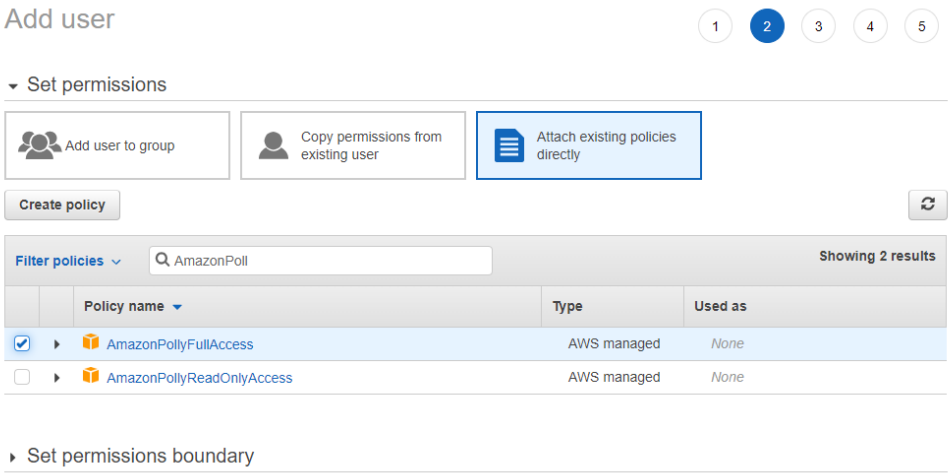
Add users:

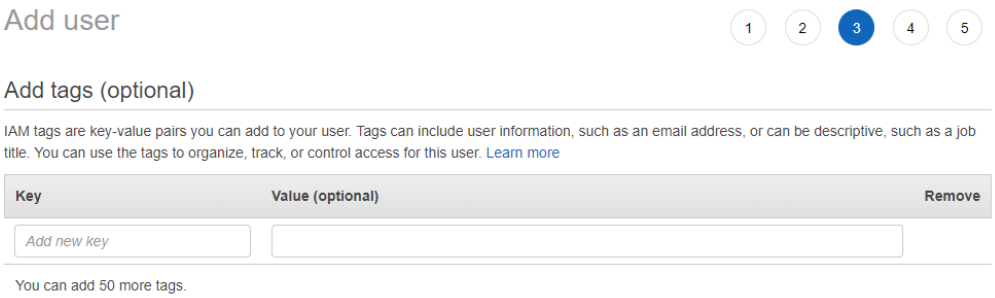




2. Now you can configure your user to give permission to access Amazon Polly.

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





Add user

1 2 3 4 5

Review

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

User details

User name	anna
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

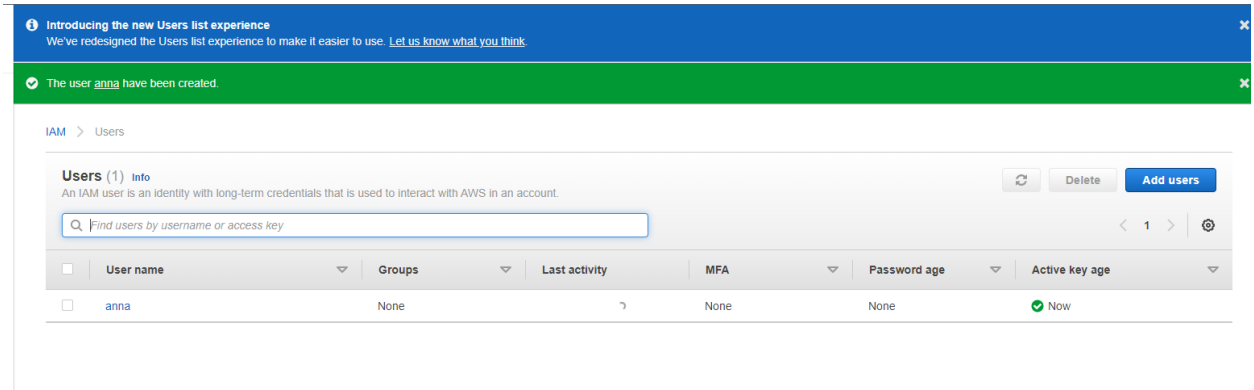
No tags were added.

Cancel Previous Create user

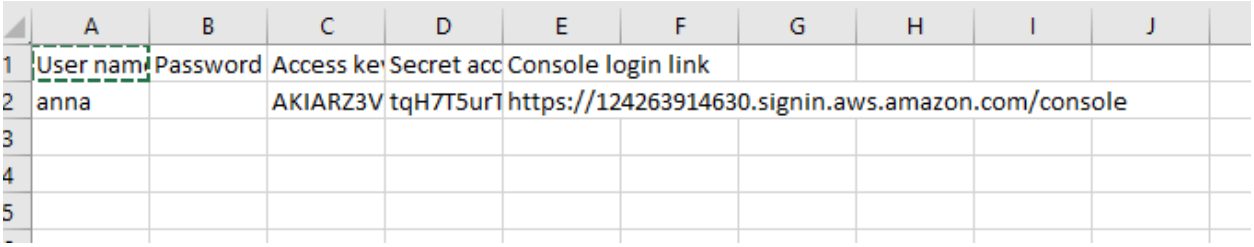
➔ User created successfully, download the .csv file for records

	A	B	C	D	E	F	G	H	I	J
1	User nam	Password	Access ke	Secret acc	Console login link					
2	anna		AKIARZ3V	tqH7T5urT	https://124263914630.signin.aws.amazon.com/console					
3										
4										
5										

User name	Password	Access key ID	Secret access key	Console login link
anna		AKIARZ3VU3SDHKYPLZOL	tqH7T5urTDPZZf9D3EizPcXvNZPMliBeP+wZNfMT	https://124263914630.signin.aws.amazon.com/con



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.



	A	B	C	D	E	F	G	H	I	J
1	User nam	Password	Access ke	Secret acc	Console login link					
2	anna		AKIARZ3V	tqH7T5urT	https://124263914630.signin.aws.amazon.com/console					
3										
4										
5										

User name	Password	Access key ID	Secret access key	Console login link
anna		AKIARZ3VU3SDHKYPLZOL	tqH7T5urTDPZZf9D3EizPcXvNZPMliBeP+wZNfMT	https://124263914630.signin.aws.amazon.com/con

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/
```

Install node.js and npm:

```
$ sudo apt update

$ sudo apt install nodejs

$ node –v

$ sudo apt install npm

$ npm install aws-sdk –save
$ npm init
```

```
anna@raspberry: ~/ml
File  Edit  Tabs  Help
anna@raspberry:~ $ mkdir ml
anna@raspberry:~ $ cd mk/
bash: cd: mk/: No such file or directory
anna@raspberry:~ $ ls
Bookshelf  Documents  ml     Pictures  Templates
Desktop    Downloads  Music  Public    Videos
anna@raspberry:~ $ cd ml/
anna@raspberry:~/ml $ npm init
bash: npm: command not found
anna@raspberry:~/ml $ sudo apt update
Hit:1 http://deb.debian.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian-security bullseye-security InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Hit:4 http://archive.raspberrypi.org/debian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
2 packages can be upgraded. Run 'apt list --upgradable' to see them.
anna@raspberry:~/ml $ S
```

```
anna@raspberry:~/ml $ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (ml)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/anna/ml/package.json:
```

anna@raspberry: ~/ml

File Edit Tabs Help

```
uilding dependency tree... Done
eading state information... Done
 packages can be upgraded. Run 'apt list --upgradable' to see them.
nna@raspberry:~/ml $ sudo apt install nodejs
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
he following package was automatically installed and is no longer required:
 sse3-support
se 'sudo apt autoremove' to remove it.
he following additional packages will be installed:
 libc-ares2 libjs-highlight.js libnode72 nodejs-doc sse2-support
uggested packages:
 npm
he following NEW packages will be installed:
 libc-ares2 libjs-highlight.js libnode72 nodejs nodejs-doc sse2-support
 upgraded, 6 newly installed, 0 to remove and 2 not upgraded.
eed to get 11.7 MB of archives.
fter this operation, 49.6 MB of additional disk space will be used.
o you want to continue? [Y/n] y
et:1 http://deb.debian.org/debian bullseye/main i386 sse2-support i386 6 [8,544
B]
et:2 http://deb.debian.org/debian bullseye/main i386 libc-ares2 i386 1.17.1-1+d
b11u1 [106 kB]
```

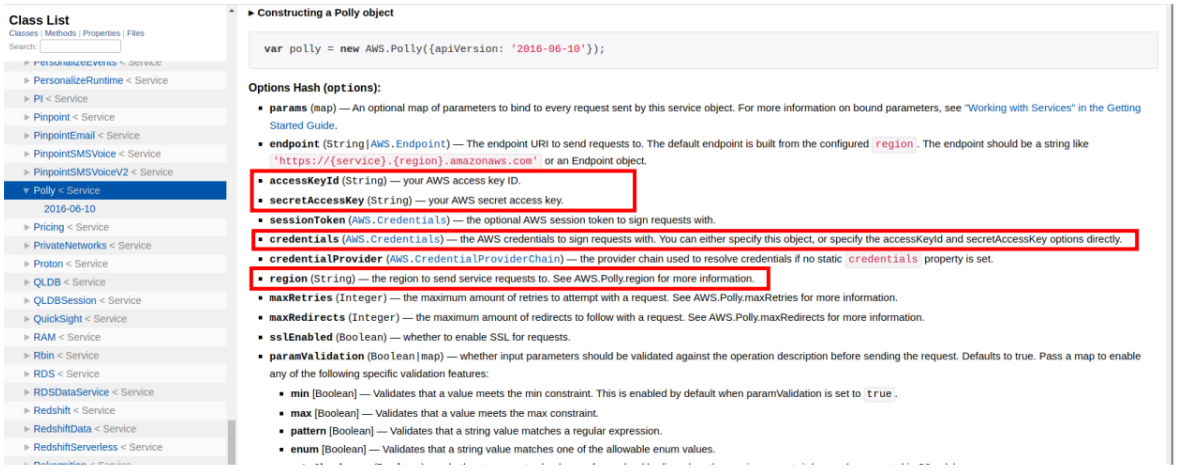
```
anna@raspberry:~/ml $ node -v
v12.22.12
anna@raspberry:~/ml $
```

```
anna@raspberry: ~/ml
File  Edit  Tabs  Help
Processing triggers for man-db (2.9.4-2) ...
anna@raspberry:~/ml $ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (ml)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/anna/ml/package.json:

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

```
anna@raspberry:~/ml $ npm install aws-sdk --save
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchP
arams API instead.

added 30 packages, and audited 31 packages in 6s

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

found 0 vulnerabilities
anna@raspberry:~/ml $
```

Project folder ready!

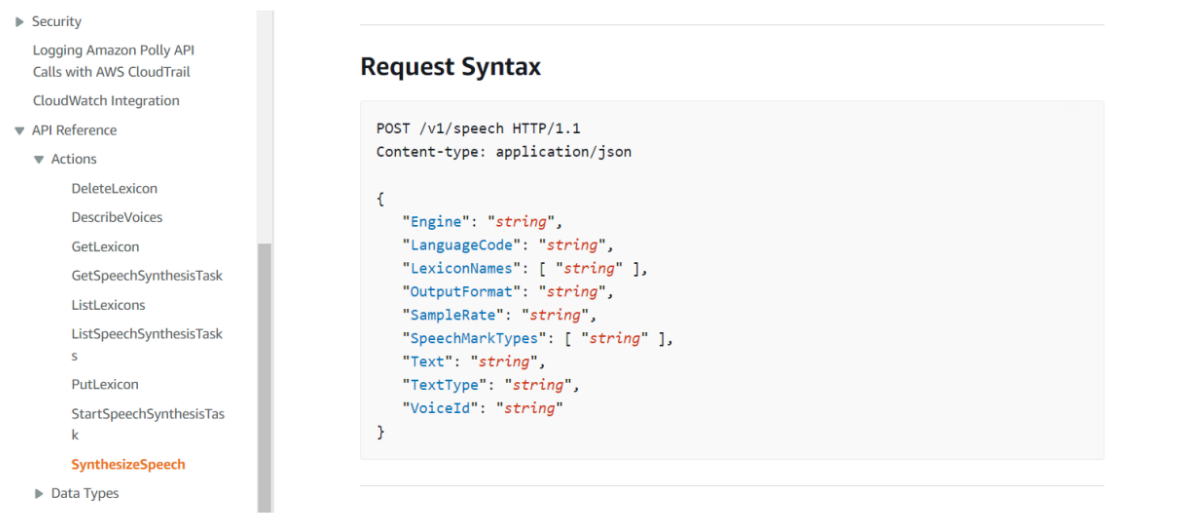
5. We will use the Polly object to access AWS Polly from Node.js. You can read more information about the Polly object on https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Polly.html.
   We pass our AWS access key ID to perform AWS authentication.



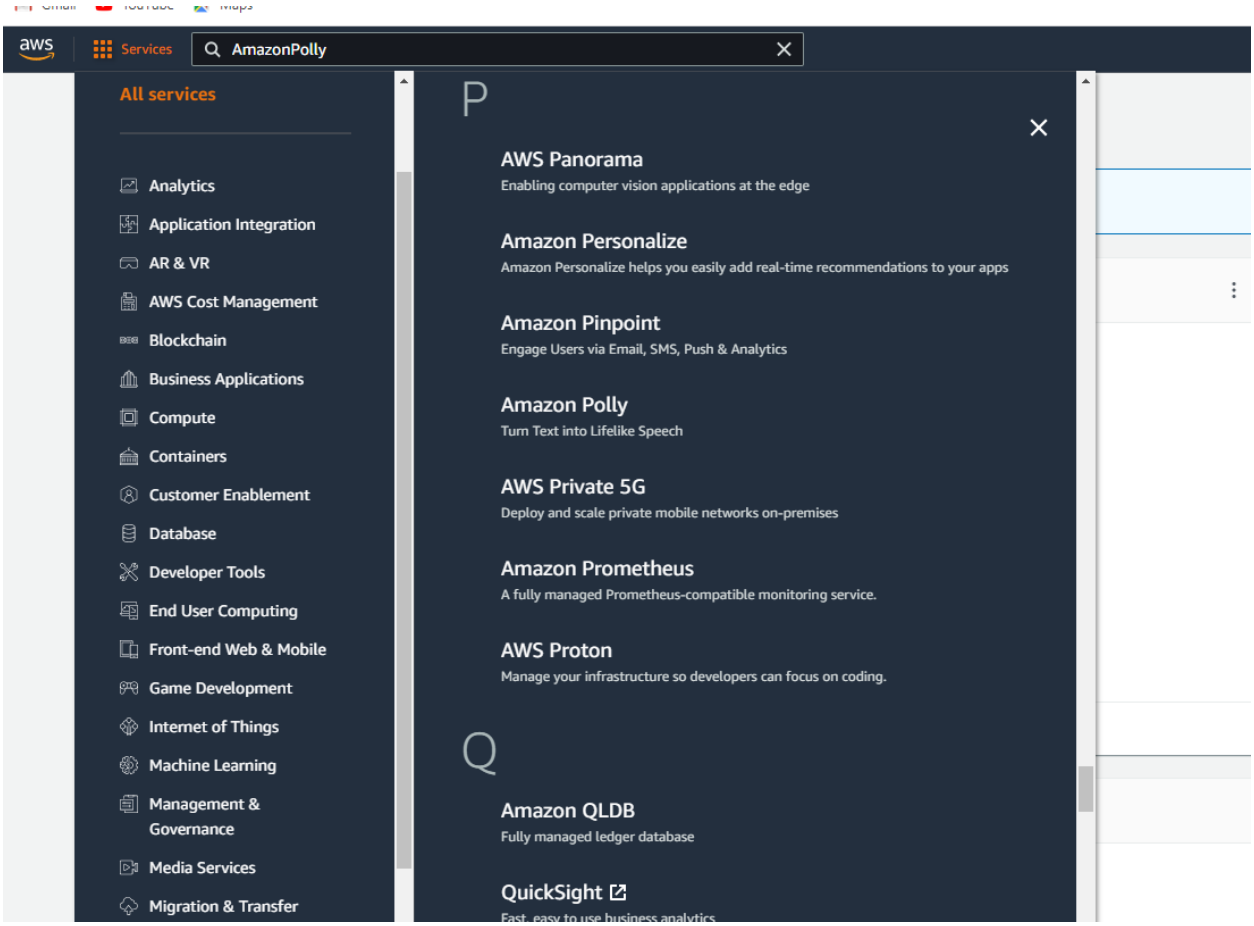
6. To convert from text-to-speech, we can call Polly.synthesizeSpeech(). From this process, we can save the result into an MP3 file.

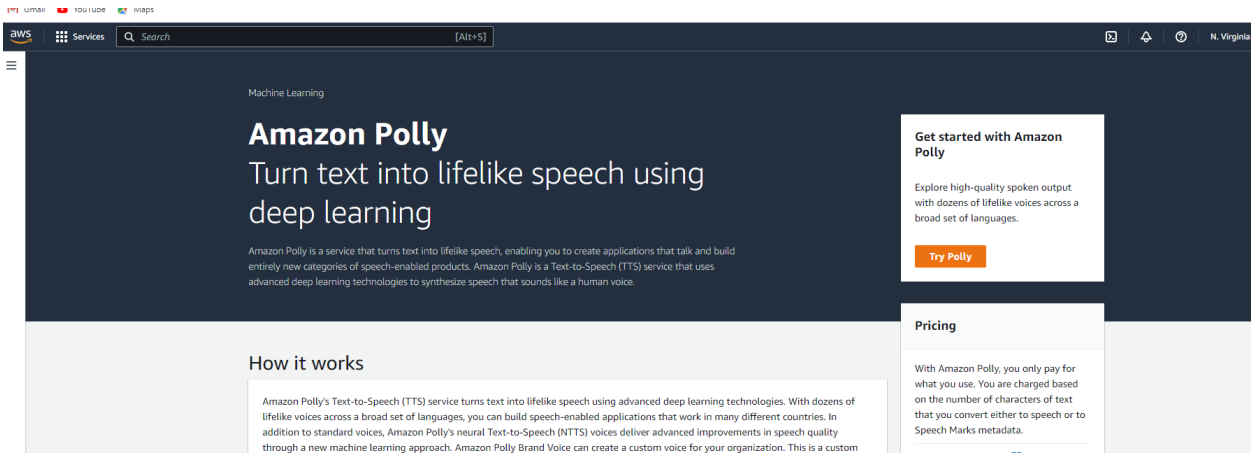
➔ Go to Amazon Polly Documentation to learn more about the Request body structure.

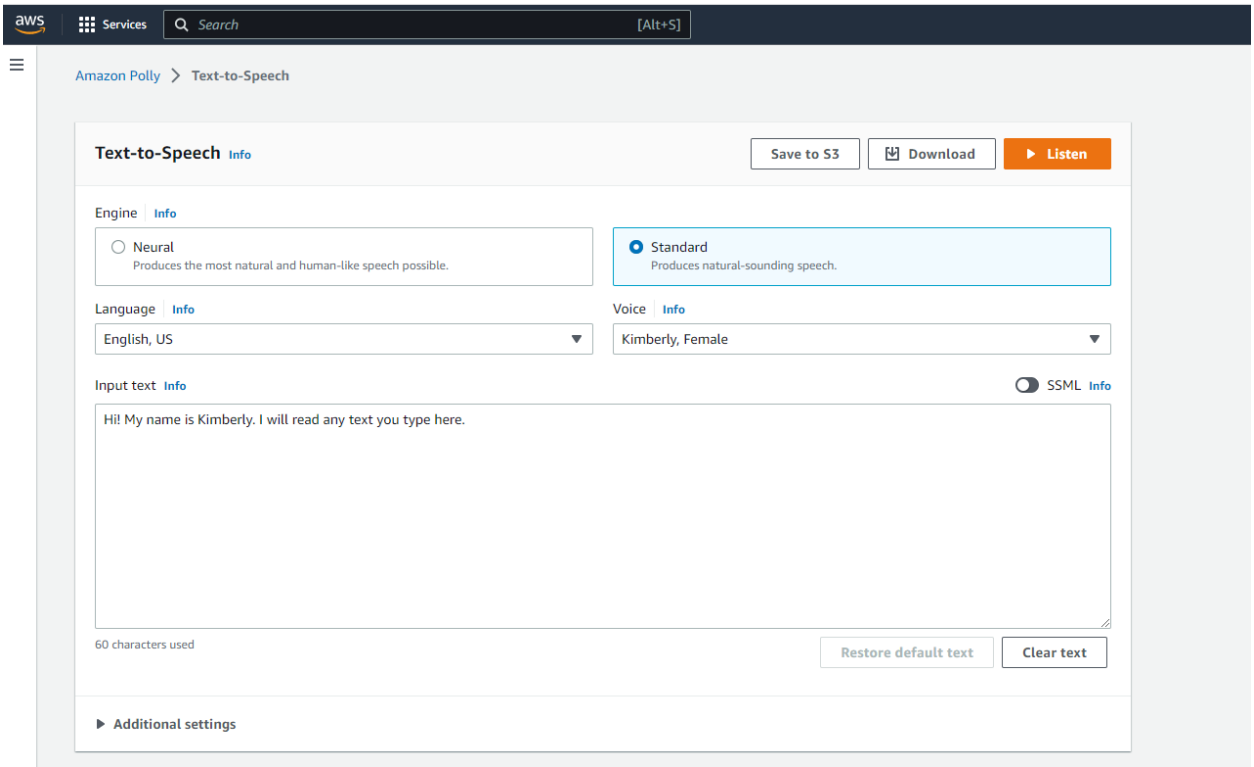
https://docs.aws.amazon.com/polly/latest/dg/API_SynthesizeSpeech.html



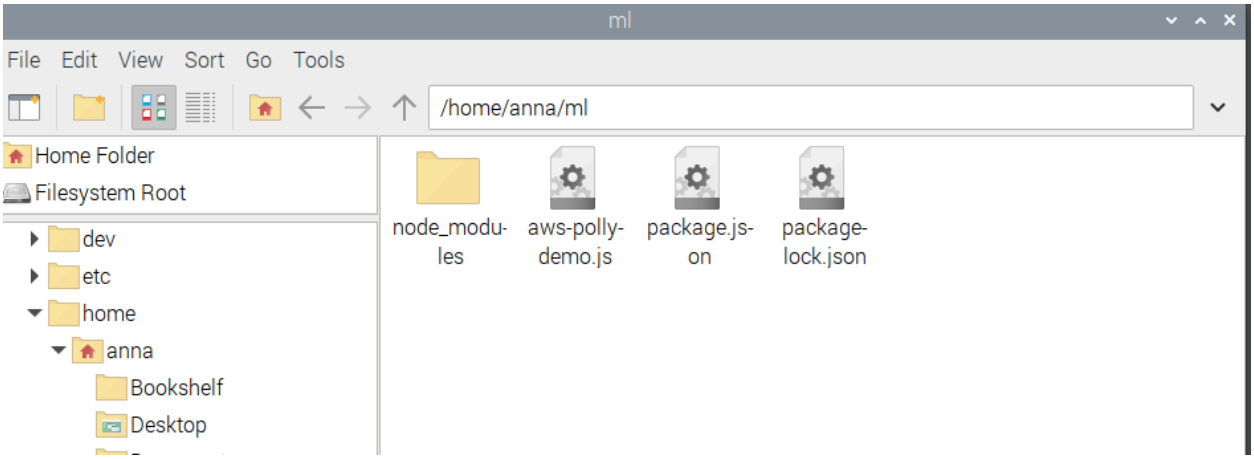
Go to aws console -> Services -> All Services -> Amazon Polly -> Try Polly Select and try your desired VoiceId, languages and so on


aws
Services
AmazonPolly
All services
Analytics
Application Integration
AR & VR
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
Media Services
Migration & Transfer
P
AWS Panorama
Enabling computer vision applications at the edge
Amazon Personalize
Amazon Personalize helps you easily add real-time recommendations to your apps
Amazon Pinpoint
Engage Users via Email, SMS, Push & Analytics
Amazon Polly
Turn Text into Lifelike Speech
AWS Private 5G
Deploy and scale private mobile networks on-premises
Amazon Prometheus
A fully managed Prometheus-compatible monitoring service.
AWS Proton
Manage your infrastructure so developers can focus on coding.
Q
Amazon QLDB
Fully managed ledger database
QuickSight


aws
Services
Search
[Alt+S]
N. Virginia
Machine Learning
Amazon Polly
Turn text into lifelike speech using deep learning
Amazon Polly is a service that turns text into lifelike speech, enabling you to create applications that talk and build entirely new categories of speech-enabled products. Amazon Polly is a Text-to-Speech (TTS) service that uses advanced deep learning technologies to synthesize speech that sounds like a human voice.
Get started with Amazon Polly
Explore high-quality spoken output with dozens of lifelike voices across a broad set of languages.
Try Polly
Pricing
With Amazon Polly, you only pay for what you use. You are charged based on the number of characters of text that you convert either to speech or to Speech Marks metadata.
How it works
Amazon Polly's Text-to-Speech (TTS) service turns text into lifelike speech using advanced deep learning technologies. With dozens of lifelike voices across a broad set of languages, you can build speech-enabled applications that work in many different countries. In addition to standard voices, Amazon Polly's neural Text-to-Speech (NTTS) voices deliver advanced improvements in speech quality through a new machine learning approach. Amazon Polly Brand Voice can create a custom voice for your organization. This is a custom



6. Let's create a file, aws-polly-demo.js.
   ➔ Create and edit aws-polly-demo.js



➔ Change the credentials and Input values

```
const Polly = new AWS.Polly(
{ accessKeyId: 'xxxxx',
secretAccessKey: xxxxx',
signatureVersion: 'v4',
```

region: 'us-east-1' });

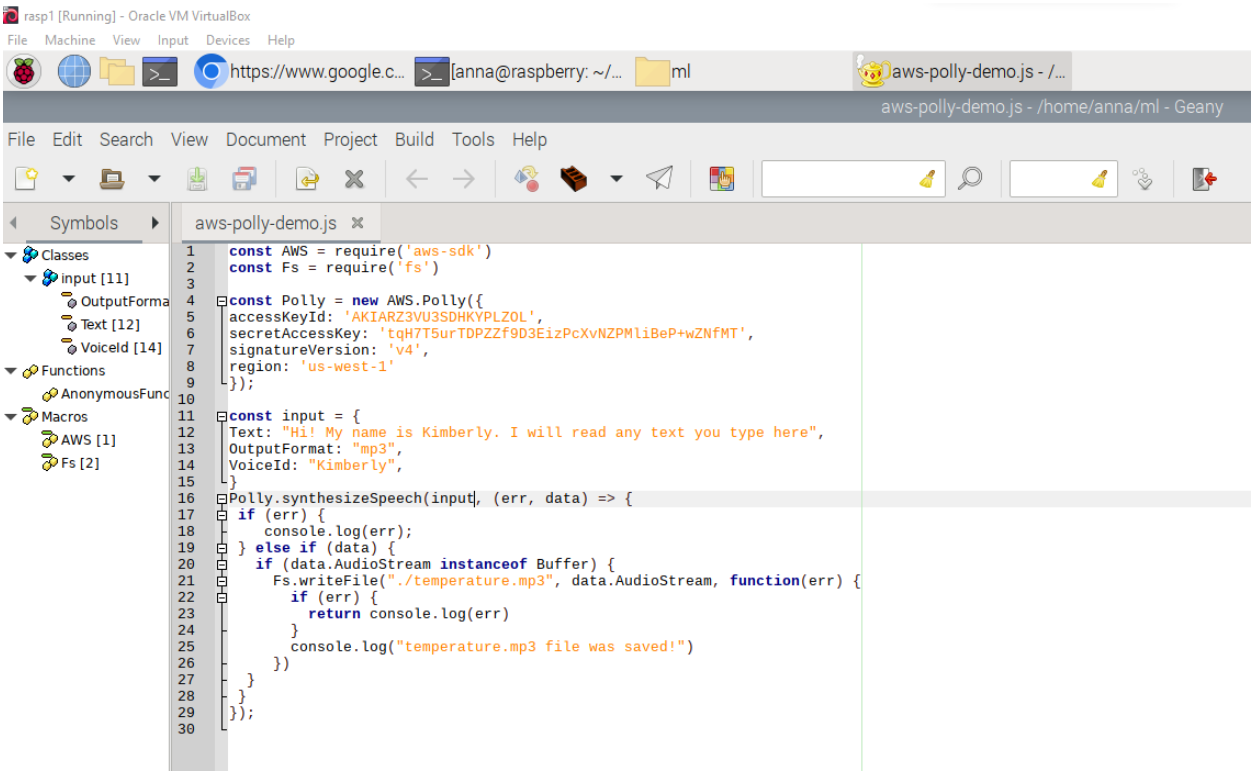
const input = {

Text: "Hello, this is a test for temperature records",

OutputFormat: "mp3",

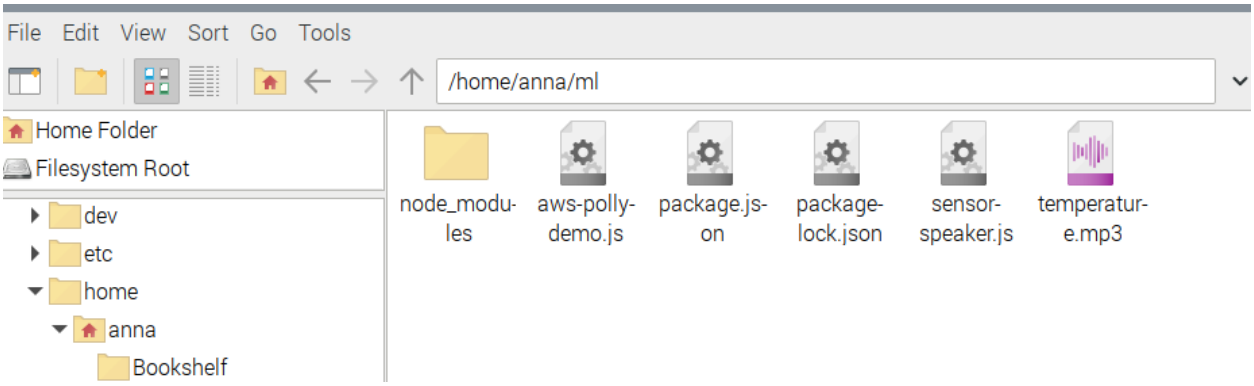
VoiceId: "xxx",

}



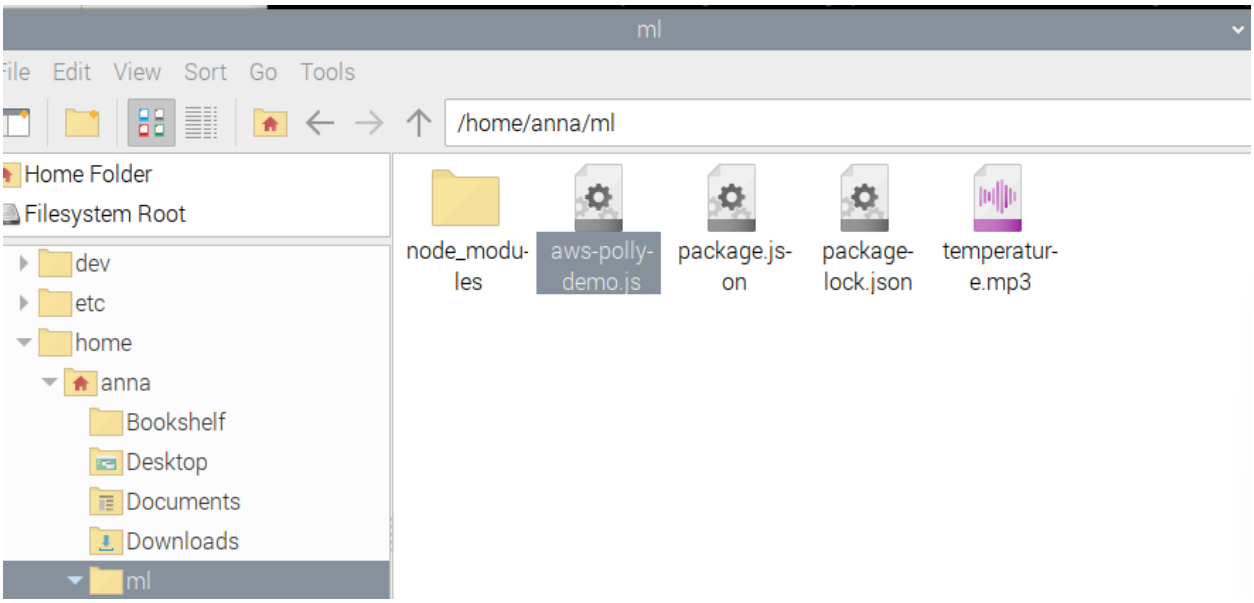
8. Save this program and run it using the following command:

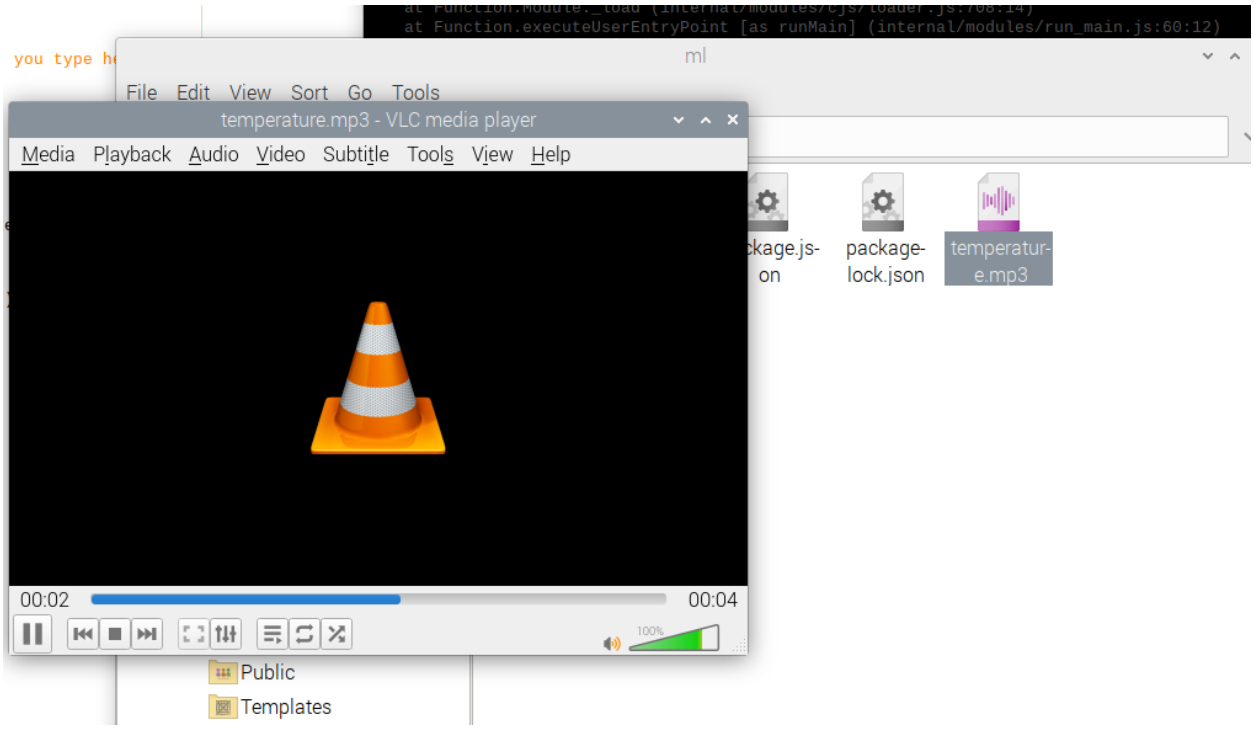
$ node aws-polly-demo.js




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

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:





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

```
anna@raspberry:~/ml $ npm install speaker
npm ERR! code 1
npm ERR! path /home/anna/ml/node_modules/speaker
npm ERR! command failed
npm ERR! command sh -c node-gyp rebuild
npm ERR! make: Entering directory '/home/anna/ml/node_modules/speaker/build'
npm ERR!   CC(target) Release/obj.target/output/deps/mpg123/src/output/alsa.o
npm ERR! make: Leaving directory '/home/anna/ml/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/anna/ml/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

node-speaker

Output PCM audio data to the speakers

Node CI failing

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
```

```
anna@raspberry:~/ml $ 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 (786 kB/s)
Selecting previously unselected package libasound2-dev:i386.
(Reading database ... 169413 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) ...
anna@raspberry:~/ml $
```

Package installation successfully!

12. Now we modify our previous program to play text-to-speech streaming into node-speaker library.

Install speaker:

$ npm install speaker

Create the sensor-speaker.js file

Code source:

[Developing a program for Amazon Polly (sfbu.edu)]

➔ Create and modify sensor-speaker.js file

```
// Create a 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',
}
```

rasp1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

https://www.google.c... [anna@raspberry: ~/... ml sensor-speaker.js - /h...

sensor-speaker.js - /home/anna/ml - Geany

File Edit Search View Document Project Build Tools Help

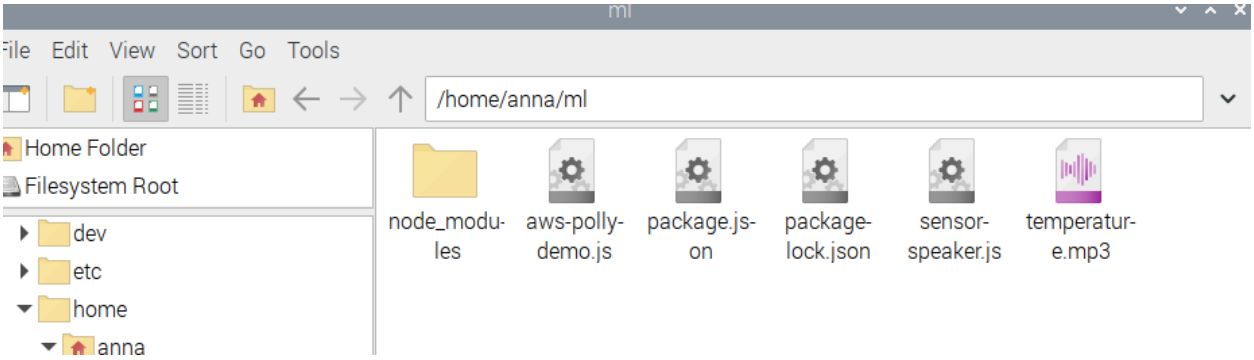
Symbols

Classes
params [24]
OutputForma
Text [25]
VoiceId [27]
Macros
AWS [1]
Player [16]
Speaker [3]
Stream [2]

aws-polly-demo.js sensor-speaker.js

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



// Create an Polly client
const Polly = new AWS.Polly({
accessKeyId: 'AKIARZ3VU3SDHKYPLZOL',
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)
  }
 }
});
```

```
anna@raspberry:~/ml $ npm install speaker

added 8 packages, and audited 39 packages in 3s

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

found 0 vulnerabilities
anna@raspberry:~/ml $ node sensor-speaker.js
anna@raspberry:~/ml $ ls
aws-polly-demo.js  node_modules  package.json  package-lock.json  sensor-speaker.js  temperature.mp3
anna@raspberry:~/ml $
```



Successfully listen the text input!