Cloud Resources Document

S6 Software Engineering 4207734 Matei-Cristian Mitran Fontys Eindhoven 08.03.2023

1. Introduction

This document serves as an insightful guide detailing all the cloud services from Google Cloud Platform(GCP) that are utilized in the project. It is designed to offer a view of how these services are used and their configuration specifics.

Included in this document, you will find an overview of each service, its purpose in the project, the configuration details and notable nuances that are unique to the project YouSound. This document is invaluable for transferability purposes.

2. Cloud Services

GCP Virtual Machines

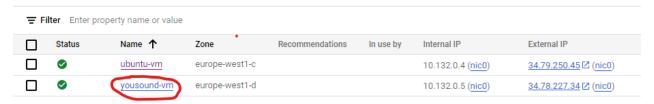
Ubuntu 20.4 LTS VM

Usage

The Ubuntu 20.4 LTS VPS serves as a testing environment in which the application is deployed via the pipeline on the testing branch.

- Docker is installed, along with Docker-Compose
- A directory called dc is called at the path /etc/dc
- A docker-compose.yml file is created that contains the services needed for deployment
- Allows only certain IPv4 addresses to ensure security
- Allows HTTP insecure connections

VM instances

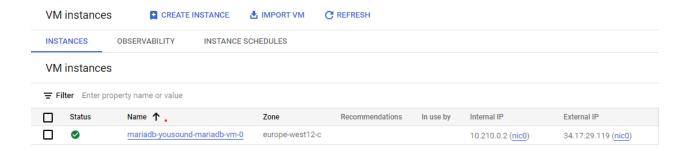


MariaDB VM

Usage

The MariaDB VM acts as a virtual environment for the deployment of a MariaDB server the contains the music database.

- MariaDB is installed on the Linux Debian VM
- A root user with a secure password is generated
- Allows only certain IPv4 addresses to ensure security
- Allows HTTP insecure connections
- Created Music database with Like, Song, Playlist and Album tables

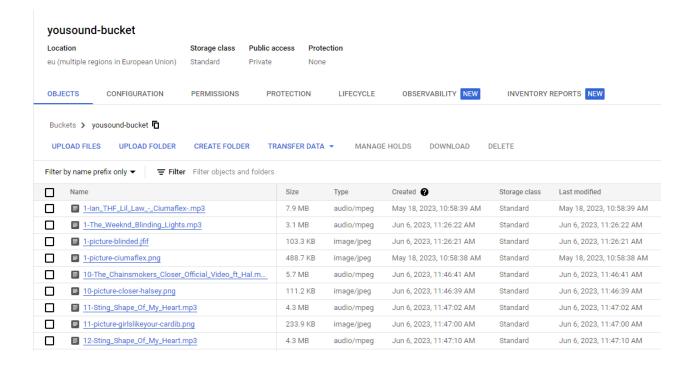


GCP Storage Bucket

Usage

There is a Google Cloud Storage Bucket that is responsible for storing the pictures and mp3 files of music-related content like songs, playlists, and albums.

- The bucket is private so only authorized users can access the URL's
- · Uniform access control
- No object versioning



Cloud Functions

Music Analytics Function

Usage

The use of this function is to get the total playtime of a song in order to provide analytics about the song. The total playtime is calculated by multiplying the duration of the song with the plays.

Configuration

- Allows HTTP
- Allows unauthorized invocations
- Runtime: Node.js 18

```
1
     exports.songAnalytics = async (req, res) => {
 2
         // Extract song data from request
3
         const song = req.body;
5
         // Check if all required data is present
6
         if (!song.duration) {
7
              res.status(400).send('Missing song data');
8
              return;
9
10
11
         // Calculate total play time and like-to-play ratio
12
         const totalPlayTime = song.duration * song.plays;
13
14
         // Return analytics data
15
         res.status(200).send({
16
              totalPlayTime: totalPlayTime
17
         });
     };
```

Token Validator Function

Usage

The use of this function is to validate a JWT token provided in the body of the request. It is used via a circuit breaker that in the api gateway when the security service is down and the endpoint to validate the token is not available, the request is sent to this serverless function, therefore ensuring availability.

- Allows HTTP
- Allows unauthorized invocations
- Runtime: Node.js 18
- · Dependencies:
 - o express: "^4.17.1"
 - o cors: "^2.8.5"
 - o jsonwebtoken: "^8.5.1"

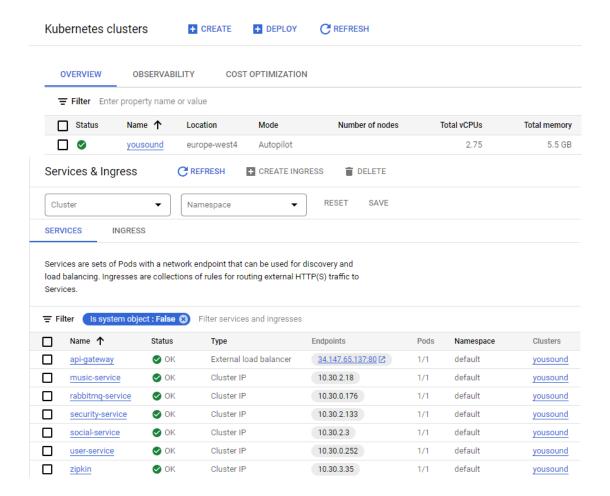
```
const express = require('express');
     const cors = require('cors');
 2
 3 const jwt = require('jsonwebtoken');
     const app = express();
     app.use(cors());
     app.post('/api/auth/validate', async (req, res) => {
 8
 9
         const token = req.body.token;
10
          // Validate your token here.
11
12
             const publicKey = `----BEGIN PUBLIC KEY-----
13
14 MIIBIjANBgkqhkiG9w@BAQEFAAOCAQ8AMIIBCgKCAQEAmJrc23N49AOWJwEL+445
    LLiLje+82smpcF4g4A8++KikDCAuGKplpAIXgffnAceyCTVHtCCeegFEZ+3Bb+bq
15
      fdSmGI2BSXhs1uMXrksj365MoLTQCP8pQ0KM+CQZvNRDte5XjK41Ny7vp624WjoB
17
     WU+01X7eFGnjnKuCc3+pItD+TIppGS1Z1lusUNWSqYhxDM2PGMKgb8X0c1bvFtru
    RcMvD02GWaNfyT+2zLndcA4B4nxXFi0O2dB78H+wIKDLRIgf3bHw6gLLN1ChPxHL
18
19
     TFGbZabsXKD72wLMMJLJcqJ5p9/yjE2e1R16YBIMe45KY1O4AVmBcfc5wNBu8ruC
20
      -----END PUBLIC KEY-----';
21
22
             const decodedToken = jwt.verify(token, publicKey, { algorithms: ['RS256'] });
23
24
25
             // Check if the token is expired.
             const currentTimestamp = Math.floor(new Date().getTime() / 1000);
             if (decodedToken.exp < currentTimestamp) {</pre>
27
28
                  res.status(401).send({ message: "Unauthorized: Token expired" });
                 // Return a default user if the token is valid
30
31
                 res.send(true);
33
          } catch (err) {
34
              res.status(401).send({ message: "Unauthorized: Invalid token", error: err });
35
36
     });
37
    exports.app = app;
```

Google Kubernetes Engine

Usage

Google Kubernetes Engine (GKE) provides a managed environment for deploying, scaling, and managing containerized applications. It offers automated scaling and rolling updates reducing time spent on infrastructure management. Plus, it seamlessly integrates with other Google Cloud services.

- Cloud Monitoring
- Cluster Location: europe-west4
- Service Address Range: /22
- Cluster default Pod Address Range: /17
- Public Cluster



Notable Google APIs

Cloud Logging API

Used for the monitoring of all the cloud services described.

Cloud Monitoring API

Used for the monitoring of all the cloud services described.

Cloud Natural Language API

Used for evaluating if the field "content" of entities in the social service are harmful or spam.