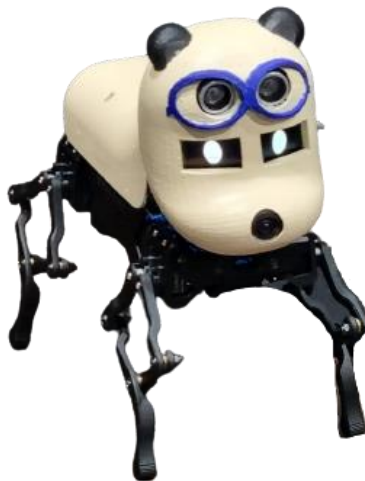


BoBi

관찰형 반려 케어 로봇 보비

보비[명] 보조하며 돌봄



Porting Manual

2022 SSAFY 공통 프로젝트

서울캠퍼스 2반 8팀

장명근 곽다원 김동원 신선영 이승훈 정재훈

목차

I. **HW** Porting Manual

1. 기본 세팅
2. 프로젝트 세팅
3. 모듈 세팅
4. 음성 인식 세팅
5. MQTT 통신 세팅
6. Video Streaming
7. S3 Access Key 등록
8. File Structure

II. **Web** Porting Manual

1. 기술 스택 (핵심 기능)
 - 1-1. React Frontend
 - 1-2. Django Backend
2. React Frontend 빌드
3. Django Backend 빌드
 - 3-1. MySQL DB 설정
 - 3-2. React Frontend와 연결
4. EC2 배포
 - 4-1. https 설정
5. FE 추가 설정
 - 5-1. Web to S3, S3 to Web Authorization
 - 5-2. Google Login (Client Side)
6. .gitignore 파일 정보
7. File Structure

I . HW Porting Manual

HW(robot)

- 로봇에서는 영상 인식, 음성 인식, 센서 등은 라즈베리파이에서 담당하고 모터, LED, 부저는 ESP32에서 담당
- 로봇 사용을 위해 기본 세팅을 진행 후 프로젝트 세팅 진행
- ESP32를 먼저 키고 라즈베리파이를 켜야 정상동작
- 자세한 내용, version 등은 [wiki](#) 혹은 각 기능의 README.md, requirements.txt 참고

기본 세팅

How to set RPI

1. 라즈베리파이에 os가 설치된 sd카드 연결
2. 원격 접속(VNC, mobaxterm 등 참고) 혹은 HDMI 케이블을 직접 연결하여 라즈베리파이 접속
3. 라즈베리파이 터미널에서 입력(로봇 기본 dependencies 설치)

```
$ sudo git clone https://github.com/waveshare/WAVEGO.git  
$ sudo python3 WAVEGO/RPi/setup.py
```

4. Completed 가 뜨고 라즈베리파이가 재부팅 되면 성공

[영상 참고](#)

How to set Arduino(ESP32)

1. Arduino IDE를 개인 PC에 다운 받기
2. ESP32 개인 PC에 유선으로 연결
3. file → Preferences → Additional Boards Manager URLS 에
`https://dl.espressif.com/dl/package_esp32_index.json` 입력 → OK 입력
4. IDE 다시 시작
5. Tools → Board → Boards에서 ESP32 검색 & Install 클릭
6. 라이브러리 설치
 - Tools → Manage Libraries에서 아래 라이브러리 검색하여 설치(Install 혹은 Update)
 - ArduinoJson
 - AdafruitSSD1306
 - AdafruitPWM Servo Driver Library
 - ICM20948 WE
 - INA219 WE
 - AdafruitNeoPixel
7. 로봇 데모코드를 다운받아 압축 풀기
8. WAVEGO/Arduino/WAVEGO/WAVEGO.ino를 Arduino IDE에서 run
9. Tools → Port에서 아두이노가 연결되어 있는 포트 선택
10. Tools → Boards → ESP32 Arduino → ESP32 Dev Module 선택
 - configure 세팅 예시
Upload Speed: "921600"
CPU Frequency: "240MHz(WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB(32Mb)"
" Partition Scheme: "Huge APP(3MB No OTA/1MB SPIFFS)"
PSRAM: "Enabled""
 - Tools → PSRAM → Enabled 로 되어 있어야 함
11. Upload 버튼 클릭

로봇 데모 ESP32 참고

🔗 프로젝트 세팅

- 개인 PC에서 코드 가져오기
- git bash에서

```
$ git clone https://lab.ssafy.com/s07-webmobile3-sub2/S07P12A208.git
```

- 라즈베리파이에 코드 넣기
- clone한 dir에서 HW/VOICE_DETECTION 내에 있는 파일(version_0 dir 제외)를 모두 라즈베리파이의 ~/WAVEGO/RPi 내로 옮김

🔧 모듈 세팅

사용할 모듈

- OLED 128x64 I2C 지원 2EA
- DHT11 온습도 센서
- MQ-135 Gas 센서
- TTP223 정전식 터치센서
- HC-SR04 초음파 센서
- 3.5mm 스피커
- USB 마이크 2개
- 모두 3.3V 사용

Pin Map

표 1 pin map

센서	센서 pin	RPI pin
Touch	Data	GPIO 4
Gas	Data	GPIO 17
DHT11	Data	GPIO 18
초음파	Data1	GPIO 27
초음파	Data2	GPIO 22
OLED	SDA	GPIO 2
OLED	SCL	GPIO 3
VCC		Board 1
GND		Board 14

- 모든 센서의 VCC, GND는 빵판을 이용하여 표에 표기된 VCC, GND와 연결
- OLED 2개의 SDA, SCL은 각각 빵판을 이용하여 표에 표기된 SDA, SCL에 연결(동일한 신호를 OLED 2개에서 받음)
- 3.5mm 스피커는 3.5mm 잭에 연결하여 사용
- USB 마이크 2개는 라즈베리파이 USB 2.0에 각각 꽂아 사용

1. I2C 통신 활성화

```
$ sudo raspi-config
```

3. Interface Options → P5. I2C

2. 라이브러리 설치

```
$ git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
```

```
$ cd Adafruit_Python_SSD1306
```

```
$ sudo python3 setup.py install
```

```
$ sudo pip3 install Adafruit_BBIO
```

```
$ pip3 install adafruit-circuitpython-dht
```

```
$ sudo apt-get install libgpiod2
```

```
$ python3 -m pip install mysql-connector
```

```
$ python3 -m pip install mysql-connector-python
```

3. 센서 연결 테스트

```
$ python3 ~/WAVEGO/RPi/mysql_sensing.py
```

🔊 음성 인식 세팅

- 라즈베리파이4B Debian Buster 버전(2022.07.26 기준 RPI imager의 legacy 버전)

1. [Google Speech to Text console start guide](#)를 따라 프로젝트 설정 진행

- 프로젝트 생성
- 사용자 인증 정보 페이지 상단 + 사용자 인증 정보 만들기-> 서비스 계정-> 계정 이름 설정
- 만든 서비스 계정 클릭 페이지 상단 메뉴 바의 키-> json 키 만들기 하여 json 키 다운 받기
- 라즈베리파이에 해당 키 파일 저장

2. [python 환경 세팅](#)

```
sudo apt update
```

```
sudo apt install python3 python3-dev python3-venv
```

```
sudo apt-get install wget
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
pip3 --version
```

```
cd your-project
python3 -m venv env
source env/bin/activate
```

3. google cloud speech 설치

4. client library 설치

```
pip install --upgrade google-cloud-speech
```

5. `export GOOGLE_APPLICATION_CREDENTIALS="KEY_PATH"` 를 이용하여 이전에 RPI에 넣어놓은 key 파일 등록

- 이 때 상대 경로로 하면 찾을 수 없다는 에러가 발생하므로 ****절대 경로****를 사용할 것

6. 라즈베리파이 OS 버전 이슈 해결

- `ImportError: /lib/arm-linux-gnueabi/libm.so.6: version GLIBC_2.29 not found (required by /home/pi/google_stt/env/lib/python3.7/site-packages/grpc/_cython/cygrpc.cpython-37m-arm-linux-gnueabi.so)` 발생
- Google PubSub가 Debian GLIBC 2.29를 지원하지 않는 것으로 보임
- 해결 방안으로는 아래 두 가지 존재
 - 라즈베리파이의 OS를 Ubuntu로 변경.
 - PATH를 설정한 후 grpcio를 다시 설치
 - 프로젝트 특성 상 라즈베리파이의 OS를 변경하는 것은 불가능 했으므로 두 번째 방법 사용

1. 텍스트 에디터로 `/home/pi/.bashrc` 에 아래 줄 추가

```
export PATH="$HOME/.local/bin:$PATH"
```


2. 아래 명령어 실행

```
pip uninstall grpcio
pip uninstall grpcio-status
pip install grpcio==1.44.0 --no-binary=grpcio
pip install grpcio-tools==1.44.0 --no-binary=grpcio-tools
```

3, 4번째 명령어 합쳐 실행하는데 1시간 정도 걸림

7. google stt 진행하던 대로 `/home/pi/google_stt` 내 설정된 env activate(`source env/bin/activate``)

8. `pvpорcupine, pvpорcupinedemo` 가상 환경 내 설치

```
pip3 install pvpорcupine
pip3 install pvpорcupinedemo
```

9. 라즈베리파이의 프로젝트 폴더로 이동

```
$ cd ~/WAVEGO/RPi
```

10. 필요한 라즈베리파이 설치

```
$ (env) pip install -r voice_requirements.txt
```

11. 연결된 마이크 확인

```
$ (env) porcupine_demo_mic --show_audio_devices
```

결과 예시

```
index: 0, device name: USB PnP Sound Device Analog Mono
index: 1, device name: USB PnP Sound Device Analog Mono
index: 2, device name: Monitor of Built-in Audio Analog Stereo
- USB 마이크가 2개 인식되면 잘 됨
```

12. 파일 실행(가상 환경 없어도 됨)

- 11번까지는 초기에만 진행하면 됨

```
$ ~/WAVEGO/RPi/voice.sh [마이크 index] [user id]
```

- 마이크 index는 위에서 보는 것 중 USB 마이크 0 제외 1 혹은 2로 진행하면 됨

- user id는 웹에서 받은 DB의 아이디

🌐 MQTT 통신 세팅

- WEB과 신호를 주고 받기 위해 MQTT 사용
- 라즈베리파이에 코드 넣기
- clone한 dir에서 HW/MQTT 내에 있는 파일(version_0 dir 제외)를 모두 라즈베리파이의 `~/WAVEGO/RPi` 내로 옮김

1. google stt 진행하던 대로 `/home/pi/google_stt` 내 설정된 env activate(`source env/bin/activate`)

2. 필요한 라이브러리는 `voice_requirements.txt`에서 이미 설치 완료

3. 실행

```
$ ~/WAVEGO/RPi/mqtt.sh [user id]
```

4. 테스트

- 개인 PC에서 clone 받은 dir에서 `/HW/MQTT/version_0/mqtt_subscribe_test_for_rpi.py`를 이용하여 잘 동작하는지 확인 가능

- 개인 PC에서

```
$ python mqtt_subscribe_test_for_rpi.py -op1 i7a208.p.ssafy.io
```

- 라즈베리파이 콘솔에서 forward, backward 등이 수행이 잘 되면 성공

Video Streaming

1. 라즈베리파이에 라이브러리 설치

```
$ sudo apt update
$ sudo apt full-upgrade
$ sudo apt-get install ffmpeg
```

2. 개인 구글 아이디로 유튜브 접속

3. 스트리밍 시작하여 키 얻기

4. 라즈베리파이에서 cmd 실행

```
$ ffmpeg -re -i /dev/video0 -f lavfi -i anullsrc -vb 2500k -s 1280x720 -f flv
[youtube streaming 키]
```

S3 access key 등록

1. AWS에 로그인

2. [IAM console](#)로 이동

3. My Security Credentials → Access Keys 로 이동

4. [Create New Access Key](#) 로 새로운 키 만들기

5. csv 파일 다운 받기

6. 라즈베리파이에 원격으로 접속해서 받은 키 등록

```
$ aws configure
- 개인 PC로 받은 csv 파일을 열어 적혀 있는 access key, secret key 등록
```

[AWS key 만들기](#) [key 등록](#)

🌲 File Structure

```
HW
├── DESIGN                // 3D models
├── DOCS
│   ├── ideas
│   │   └── assets
│   ├── tips
│   │   └── assets
├── Gesture_Sensor        // 터치 센서
│   ├── img
│   ├── test
│   ├── go_oled.py
│   ├── mysql_sensing.py
│   ├── oled_touch.py
│   └── sensor_pin.txt
├── MQTT                  // web & robot 통신
│   ├── version_0
│   │   └── mqtt_in_js
│   ├── mqtt.sh
│   └── mqtt_subscribe.py
├── Object_learning       // 아이 학습
│   ├── object_img
│   ├── test_img
│   └── output.xml
├── OPENCV                // 아이 인식 & following
│   ├── version_0
│   └── camera_opencv.py
├── Sensor                // 센서 세팅
│   ├── test
│   ├── version_0
│   ├── [sensor] setting_audio.md
│   ├── [sensor] setting_dht11.md
│   ├── [sensor] setting_gas.md
│   ├── [sensor] setting_oled.md
│   ├── [sensor] setting_sonic.md
│   ├── [sensor] setting_touch.md
│   └── sensor_requirements.txt
├── STREAMING             // YouTube streaming
├── VOICE_DETECTION       // 음성 인식
│   ├── img              // OLED 에 사용되는 표정
```

```

├──version_0
│   ├──Google_STT
│   │   ├──resources
│   │   └──results
│   ├──PICOVOICE
│   ├──VOICE_DETECTION
│   └──VOICE_MSSG
├──voice_data // story, key, 한글 모델
├──sensor_pin.txt
├──sensor_mysql.py
├──sensor_oled.py
├──sensor_touch.py
├──voice.py
├──voice.sh
├──voice_porcupine_custom.py
├──voice_recognition.py
├──voice_requirements.txt
├──voice_s3_mssg.py
└──voice_speaker.py
└──WAVEGO
    ├──Demo_code
    │   ├──WAVEGO_Demo_Code_(Arduino)_220128.zip
    │   └──WAVEGO_Demo_Code_(Pi)_220128.zip
    ├──version_0
    │   ├──코드 수정 후 적용.md
    │   ├──얼굴 학습 사용법 정리.md
    │   ├──WebPage_0811.h
    │   ├──WAVEGO_0811.ino
    │   ├──ServoCtrl_0811.h
    │   ├──robot_0811.py
    │   ├──[WAVEGO] code_analysis.md
    │   ├──camera_opencv_0810.py
    │   ├──[WAVEGO] 0803_code_modify.md
    │   ├──[WAVEGO] 0804_code_modify.md
    │   ├──[WAVEGO] 0809_code_modify.md
    │   └──[WAVEGO] 0811_code_modify.md
    ├──[WAVEGO] demo_code_flow.md
    └──robot.py

```

II. Web Porting Manual

1. 기술 스택 (핵심 기능)

1-1. [React Frontend](#) (bobi_frontend/package.json 참고)

- 반응형 웹 페이지 react-router-dom 6.3.0
- 구글 로그인 react-google-login 5.2.2 / gapi-script 1.2.0
- MQTT 통신 mqtt 4.3.7
- S3 업로드 react-s3 1.3.1
- 음성 재생 플레이어 react-player 2.10.1
- 센서 값 그래프 react-apexcharts 1.4.0
- 이모티콘 @fortawesome/react-fontawesome 0.2.0 / @fortawesome/free-solid-svg-icons 6.1.2

1-2. [Django Backend](#) (bobi_backend/requirements.txt 참고)

- Django Django 3.2.12
- 교차 출처 리소스 공유 django-cors-headers 3.13.0
- 더미데이터 생성 django-seed 0.3.1
- API djangorestframework 3.13.1
- MySQL DB 연동 mysqlclient 2.1.1

2. React Frontend 빌드

- 패키지 설치 bobi_backend/bobi_frontend

```
npm i package.json
```

```
[오류 발생시] npm i --force package.json
```

- 빌드 파일 생성 (Django의 index.html과 같이 사용)

```
npm run build
```

3. Django Backend 빌드

- 가상환경 설정

```
python -m venv venv
```

```
source venv/Scripts/activate
```

- 패키지 설치

```
pip install -r requirements.txt
```

3-1. MySQL DB 설정 (EC2에 MySQL 설치, 포트 및 계정명 확인 후)

- [미설치시] pip install mysqlclient
- manage.py와 같은 위치에 my_settings.py 생성

```
- DATABASES = {  
-     'default': {  
-         'ENGINE': 'django.db.backends.mysql', #1 사용할 엔진  
-         'NAME': 'bobi', #2 연동할 MySQL의 DB 이름  
-         'USER': 'pjt_bobi', #3 DB 접속 계정명  
-         'PASSWORD': 'mysql989312bobi#', #4 DB 접속 계정 PW  
-         'HOST': 'i7a208.p.ssafy.io', #5 실제 DB 주소 (EC2)  
-         'PORT': '3306', #6 포트번호  
-     }  
- }  
-  
- # SECRET_KEY: 프로젝트의 settings.py에서 추출  
- SECRET_KEY = 'django-insecure-  
- p^xm%mmc+betjp4@)lmx!35rbl^02kygw(@$u5(_bp3b$z_9a_'
```

- 기존 settings.py 수정

```
- # Database (주석 처리)
- # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
-
- # DATABASES = {
- #     'default': {
- #         'ENGINE': 'django.db.backends.sqlite3',
- #         'NAME': BASE_DIR / 'db.sqlite3',
- #     }
- # }
-
- # MySQL 과 연동
- DATABASES = my_settings.DATABASES
- SECRET_KEY = my_settings.SECRET_KEY
```

3-2. React Frontend와 연결 (CORS)

- [미설치시] pip install django-cors-headers

- settings.py 수정

```
- ALLOWED_HOSTS = [
-     "i7a208.p.ssafy.io", # 배포 과정 위해 필요
-     "127.0.0.1"
- ]
-
- INSTALLED_APPS = [
-     ...,
-     'corsheaders',
- ]
-
- MIDDLEWARE = [
-     'corsheaders.middleware.CorsMiddleware',
- ]
-
- CORS_ORIGIN_ALLOW_ALL = False
-
- CORS_ALLOW_CREDENTIALS = True
-
- CORS_ORIGIN_WHITELIST = [
-     'http://localhost:3000',
- ]
```


4. EC2 배포 (MobaXterm cli 사용) [참고] <https://nerogarret.tistory.com/>

EC2 서버 로컬 프로젝트 폴더

EC2 기본 설정

- 패키지 정보 업데이트

```
sudo apt-get update
```

- 패키지 의존성 검사 / 업그레이드

```
sudo apt-get dist-upgrade
```

- pip3 설치

```
sudo apt-get install python3-pip
```

가상환경 설정

```
python -m venv venv
```

```
source venv/Scripts/activate
```

프로젝트 파일 전송 (로컬 - EC2)

- 깃허브 개인 리포지토리 생성 (private) - 과정 생략
- 전체 프로젝트 push (.gitignore 수정 - .env, my_settings.py 등 파일 포함해야)
- 다운받을 프로젝트 폴더 생성 (ubuntu 유저 설정)

```
sudo chown -R ubuntu:ubuntu /srv/
```

- 생성한 srv 폴더에 git clone

```
cd /srv
```

```
git clone [repository]
```

- authentication 요청 시 github id와 pw 입력 (토큰 필요시 발급)

방화벽 설정

- Django 서버에 필요한 포트 오픈

```
ufw allow 8080
```

uWSGI 서버 연결 (Django와 웹서버 연결)

- uWSGI 설치

```
pip3 install uwsgi
```

- Django 프로젝트 연결

```
pip3 install uwsgi
```

```
uwsgi --http :8080 --home /home/ubuntu/venv/ --chdir /srv/bobi_backup/ -w  
bobi_backend.wsgi
```

uWSGI 설정

- manage.py와 같은 위치에 .config/uwsgi 폴더 및 파일 생성 후 push-clone

- bobi_backup/.config/uwsgi/bobi_backend.ini

```
- [uwsgi]  
- chdir = /srv/bobi_backup/  
- module = bobi_backend.wsgi:application  
- home = /home/ubuntu/venv/  
-  
- uid = ubuntu  
- gid = ubuntu  
-  
- socket = /tmp/bobi_backend.sock  
- chmod-socket = 666  
- chown-socket = ubuntu:ubuntu  
-  
- enable-threads = true  
- master = true  
- vacuum = true  
- pidfile = /tmp/bobi_backend.pid  
- logto = /var/log/uwsgi/bobi_backend/@(exec://date +%%Y-%%m-%%d).log  
- log-reopen = true
```

```
sudo mkdir -p /var/log/uwsgi/bobi_backend
```

```
sudo chown -R ubuntu:ubuntu /var/log/uwsgi/bobi_backend/
```

```
sudo /home/ubuntu/venv/bin/uwsgi -i /srv/bobi_backup/.config/uwsgi/ mysite.ini
```

nginx 연결

- nginx 설치

```
sudo apt-get install nginx
```

- nginx config 파일 수정

```
sudo vi /etc/nginx/nginx/conf
```

```
user ubuntu; (첫번째줄 user 수정)
```

- 로컬 프로젝트 폴더에 config 파일 추가 .config/nging/bobi_backend.conf

```
- listen 80;
-     server_name *.compute.amazonaws.com;
-     charset utf-8;
-     client_max_body_size 128M;
-
-     location / {
-         uwsgi_pass unix:///tmp/mysite.sock;
-         include uwsgi_params;
-     }
- }
```

- ini 파일 수정 `.config/uwsgi/bobi_backend.ini`

```
- [uwsgi]
- chdir = /srv/bobi_backup/
- module = bobi_backend.wsgi:application
- home = /home/ubuntu/myenv/
-
- uid = ubuntu
- gid = ubuntu
-
- socket = /tmp/bobi_backend.sock
- chmod-socket = 666
- chown-socket = ubuntu:ubuntu
-
- enable-threads = true
- master = true
- vacuum = true
- pidfile = /tmp/bobi_backend.pid
- logto = /var/log/uwsgi/mysite/@(exec://date +%Y-%m-%d).log
- log-reopen = true
```

- nginx 백그라운드 실행 설정 `./config/uwsgi/uwsgi.service`

```
- [Unit]
- Description=uWSGI service
- After=syslog.target
-
- [Service]
- ExecStart=/home/ubuntu/venv/bin/uwsgi -i
  /srv/bobi_backup/.config/uwsgi/bobi_backend.ini
-
- Restart=always
- KillSignal=SIGQUIT
- Type=notify
- StandardError=syslog
- NotifyAccess=all
-
- [Install]
- WantedBy=multi-user.target
```

- git push (local) / pull (server) – 생략

- uwsgi.service 파일 daemon 등록

```
sudo ln -f /srv/bobi_backup/.config/uwsgi/uwsgi.service/etc/system/uwsgi.s
ervice
```

- daemon 새로고침

```
sudo systemctl daemon-reload
```

- uwsgi 사용 설정, restart

```
sudo systemctl enable uwsgi
```

```
sudo systemctl restart uwsgi
```

- nginx 설정 파일 등록

```
sudo cp -f /srv/bobi_backup/.config/nginx/bobi_backend.conf /etc/nginx/sites-available/bobi_backend.conf
```

```
sudo ln -sf /etc/nginx/sites-available.conf /etc/nginx/sites-enabled/bobi_backend.conf
```

```
sudo rm /etc/nginx/sites-enabled/default
```

- daemon 새로고침, nginx, uwsgi 재실행

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart uwsgi nginx
```

- 포트 오픈

```
ufw allow 80
```

- Django static 연결 bobi_backend/settings.py

```
- STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

- git pull 이후 collectstatic 명령어 (가상환경 pip 설치 확인)

```
python3 manage.py collectstatic
```

- config 파일에 static 루트 추가 `./config/nginx/bobi_backend.conf`

```
- server {  
-     listen 80;  
-     server_name *.compute.amazonaws.com;  
-     charset utf-8;  
-     client_max_body_size 128M;  
-  
-     location / {  
-         uwsgi_pass unix:///tmp/mysite.sock;  
-         include uwsgi_params;  
-     }  
-  
-     location /static/ {  
-         alias /srv/django-deploy-test/static/;  
-     }  
- }
```

- conf 파일 재등록

```
sudo cp -f /srv/bobi_backup/.config/nginx/bobi_backend.conf /etc/nginx/sites-  
available/bobi_backend.conf
```

```
sudo ln -sf /etc/nginx/sites-available/bobi_backend.conf /etc/nginx/sites-  
enabled/bobi_backend.conf
```

- daemon 새로고침, nginx, uwsgi 재실행 -- 생략

4-1. https 설정 (certbot 이용)

- certbot 설치 (nginx 설치 확인)

```
sudo add-app-repository ppa:certbot/certbot
```

```
sudo apt-get update
```

```
sudo apt-get install python-certbot-nginx
```

- let's encrypt 인증서 발급

```
sudo certbot certonly --nginx -d i7a208.p.ssafy.io
```

- SSL 인증서 관련 설정

```
sudo vi /etc/nginx/sites-available/default
```

```
- server {  
-     listen 443;  
-     listen [::]:443;  
-     ssl on; #ssl 활성화  
-     server_name _;  
-  
-     ssl_certificate  
-     /etc/letsencrypt/live/i7a208.p.ssafy.io/fullchain.pem;  
-     ssl_certificate_key  
-     /etc/letsencrypt/live/i7a208.p.ssafy.io/privkey.pem;  
-  
-     location / {  
-         # First attempt to serve request as file, then  
-         # as directory, then fall back to displaying a 404.  
-         try_files $uri $uri/ =404;  
-     }  
- }
```

- nginx 재시작

```
sudo service nginx restart
```

- SSL 자동 갱신 설정

```
sudo certbot renew --dry-run
```

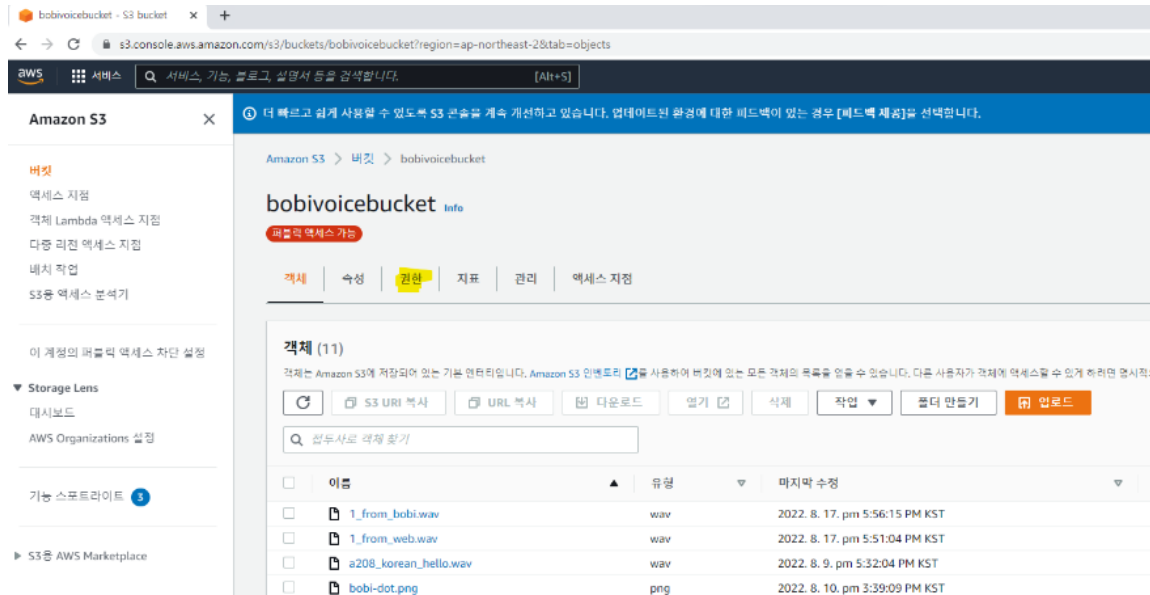
- https 포트 (443) 오픈

```
ufw allow 443
```

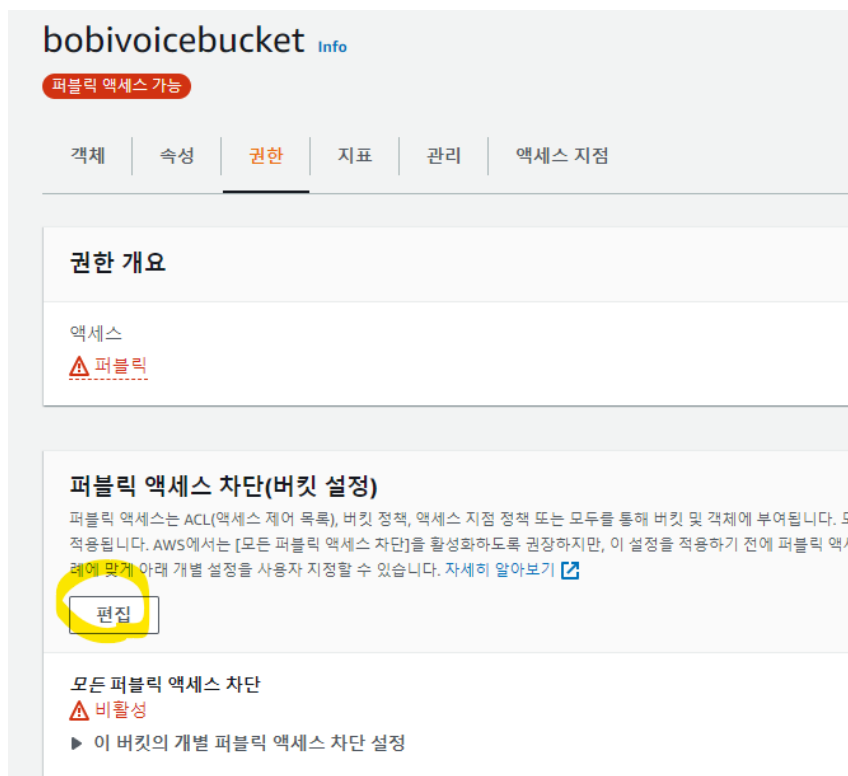
5. FE 추가 설정

5-1. Web to S3, S3 to Web Authorization

1. 버킷 내부의 권한 탭 이동



2. 퍼블릭 액세스 차단 편집



모든 차단 해제하기

퍼블릭 액세스 차단 편집(버킷 설정) Info

퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

☐ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

취소

변경 사항 저장

3. 버킷 정책 수정하기

퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

편집

모든 퍼블릭 액세스 차단

비활성

이 버킷의 개별 퍼블릭 액세스 차단 설정

버킷 정책

JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다. [자세히 알아보기](#)

편집

삭제

편집을 클릭하고 아래와 같이 입력하고 변경사항 저장

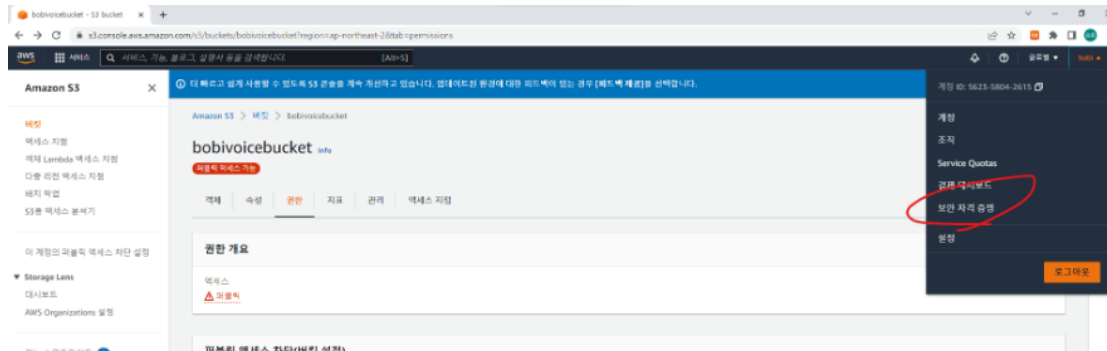
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::bobivoicebucket/*"
    }
  ]
}
```

4. CORS 정책 수정하기

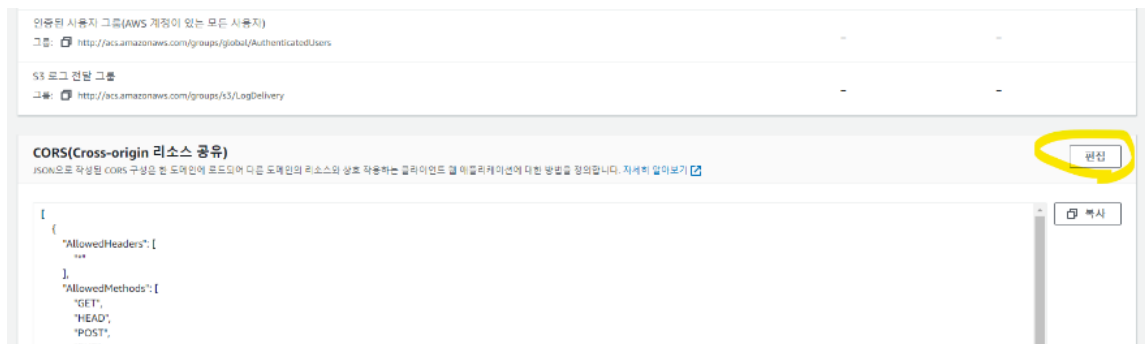
```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD",
      "POST",
      "PUT"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": [
      "Access-Control-Allow-Origin",
      "x-amz-server-side-encryption",
      "x-amz-request-id",
      "x-amz-id-2"
    ]
  }
]
```

5. 액세스 키 발급받기

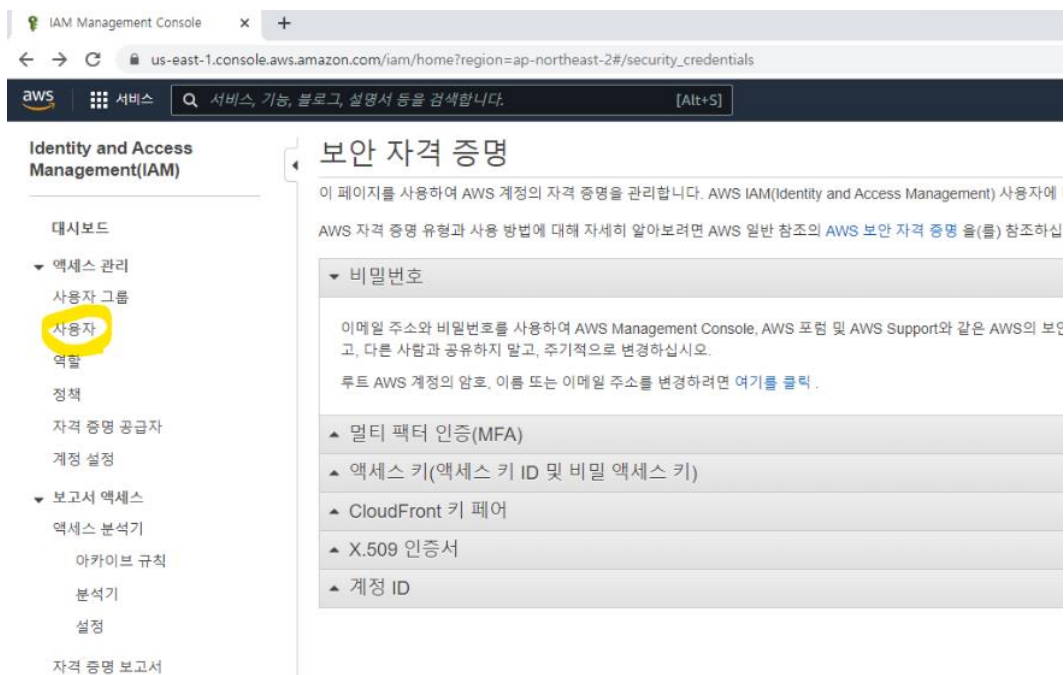
우측 상단 보안자격증명 이동



편집을 클릭하고 아래와 같이 입력하고 변경사항 저장



액세스 관리 - 사용자 탭 이동



Administrator 이동

Identity and Access Management(IAM)

새로운 사용자 목록 환경 소개
더 쉽게 사용할 수 있도록 사용자 목록 환경이 재설계되었습니다. 여러분의 생각을 알려주세요.

IAM > 사용자

사용자 (1) 정보
IAM 사용자는 계정에서 AWS와 상호 작용하는 데 사용되는 장기 자격 증명을 가진 자격 증명입니다.

사용자 이름 또는 액세스 키로 사용자 찾기

<input type="checkbox"/>	사용자 이름	그룹	마지막 활동	MFA
<input type="checkbox"/>	Administrator	Administrators	5시간 전	없음

보안 자격 증명 탭 이동

Identity and Access Management(IAM)

대시보드

- 액세스 관리
 - 사용자 그룹
 - 사용자**
 - 역할
 - 정책
 - 자격 증명 공급자
 - 계정 설정
- 보고서 액세스
 - 액세스 분석기
 - 아카이브 규칙
 - 분석기
 - 설정
 - 자격 증명 보고서
 - 조직 활동
 - SCP(서비스 제어 정책)

Q IAM 검색

AWS 계정 ID: -----

New feature to generate a policy based on CloudTrail events.
AWS uses your CloudTrail events to identify the services and actions used and generate a least privileged policy that

사용자 > Administrator

요약

사용자 ARN: arn:aws:iam::562358042615:user/Administrator
경로: /
생성 시간: 2022-08-08 16:43 UTC+0900

권한 그룹 (1) 태그 **보안 자격 증명** 액세스 관리자

Permissions policies (1 정책이 적용됨)

권한 추가

정책 이름

그룹에서 연결됨

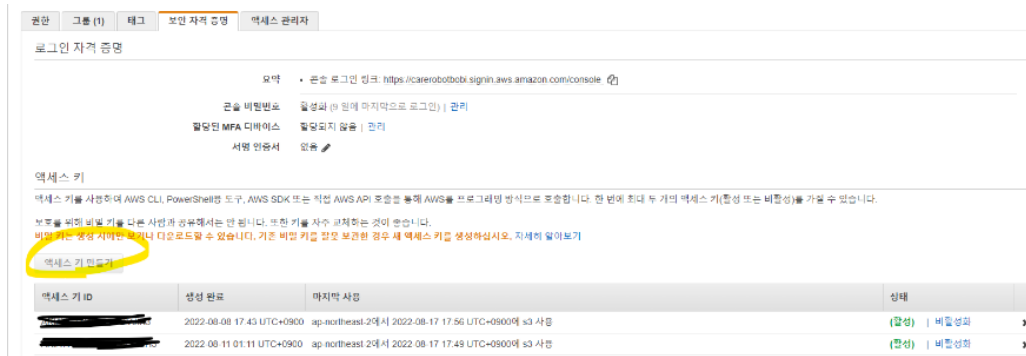
AdministratorAccess

Permissions boundary (not set)

CloudTrail 이벤트를 기반으로 정책 생성

이 프로파일에 대한 액세스 활동은 기반으로 새 정책을 생성한 다음 이 프로파일의 정책과 생성된 이 역할에 연결할 수 있습니다.

6. Access Key 만들고 Access Key, Secret Access Key 값 기록하기 (액세스 키는 2 개까지 받을 수 있어서 현재는 비활성화 상태)



로그인 자격 증명

요약 • 콘솔 로그인 링크: <https://caserobotbotbobi.signin.aws.amazon.com/console>

콘솔 비활성화 | 비밀번호 (@) 앞에 마지막으로 로그인 | 관리

활성화된 MFA 디바이스 | 활성화되지 않음 | 관리

서명 인증서 | 없음

액세스 키

액세스 키를 사용하여 AWS CLI, PowerShell, 도구, AWS SDK 또는 직접 AWS API 호출을 통해 AWS를 프로그래밍 방식으로 호출합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다.

보통을 위해 비밀 키를 다른 사람과 공유해서는 안 됩니다. 또한 키를 자주 교체하는 것이 좋습니다.

비밀 키는 생성 지점만 표시되거나 다운로드할 수 있습니다. 기존 비밀 키를 잘못 보관한 경우 새 액세스 키를 생성하십시오. 자세한 알아보기

액세스 키 만들기

액세스 키 ID	생성 완료	마지막 사용	상태
[REDACTED]	2022-08-08 17:43 UTC+0900	ap-northeast-2에서 2022-08-17 17:56 UTC+0900에 s3 사용	(활성) 비활성화 ✕
[REDACTED]	2022-08-11 01:11 UTC+0900	ap-northeast-2에서 2022-08-17 17:49 UTC+0900에 s3 사용	(활성) 비활성화 ✕

```
import { uploadFile } from "react-s3";

...

const config = {
  bucketName: S3_BUCKET,
  region: REGION,
  accessKeyId: ACCESS_KEY,
  secretAccessKey: SECRET_ACCESS_KEY,
};

...

uploadFile(file, config)
  .then((data) => console.log(data))
  .catch((err) => console.error(err));
```

7. 다운로드 할 때

1_from_bobi.wav Info

S3 URI 복사

다운로드

열기

객체 작업

속성

권한

버전

객체 개요

소유자

6864049b766fedbaf7ee07a7df9d05c946fb82ff6b7e26d5cc038f76dcb42552

AWS 리전

아시아 태평양(서울) ap-northeast-2

마지막 수정

2022. 8. 18. pm 1:38:55 PM KST

크기

656.3KB

유형

wav

키

1_from_bobi.wav

S3 URI

s3://bobivoicebucket/1_from_bobi.wav

Amazon 리소스 이름(ARN)

arn:aws:s3:::bobivoicebucket/1_from_bobi.wav

엔터티 태그(Etag)

4d8d0da9125bb3e4714a554154e17320

객체 URL

https://bobivoicebucket.s3.ap-northeast-2.amazonaws.com/1_from_bobi.wav

```
import AWS from 'aws-sdk';

...

const [source, setSource] = useState(""); // 재생할 오디오 소스 url

...

AWS.config.update({
  accessKeyId: process.env.REACT_APP_S3_ACCESS_KEY,
  secretAccessKey: process.env.REACT_APP_S3_SECRET_ACCESS_KEY,
});

const params = {
  Bucket: process.env.REACT_APP_S3_BUCKET,
  Key: `1_from_bobi.wav`, // 위 사진에서 하이라이트한 부분
};

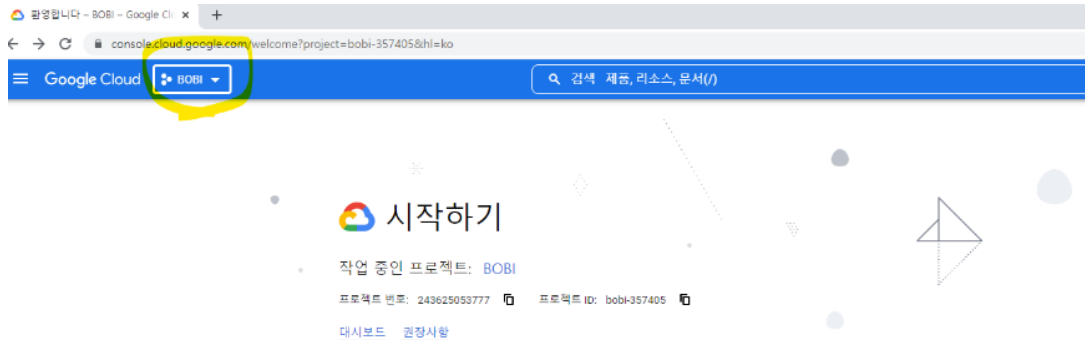
const s3 = new AWS.S3();

useEffect(() => {

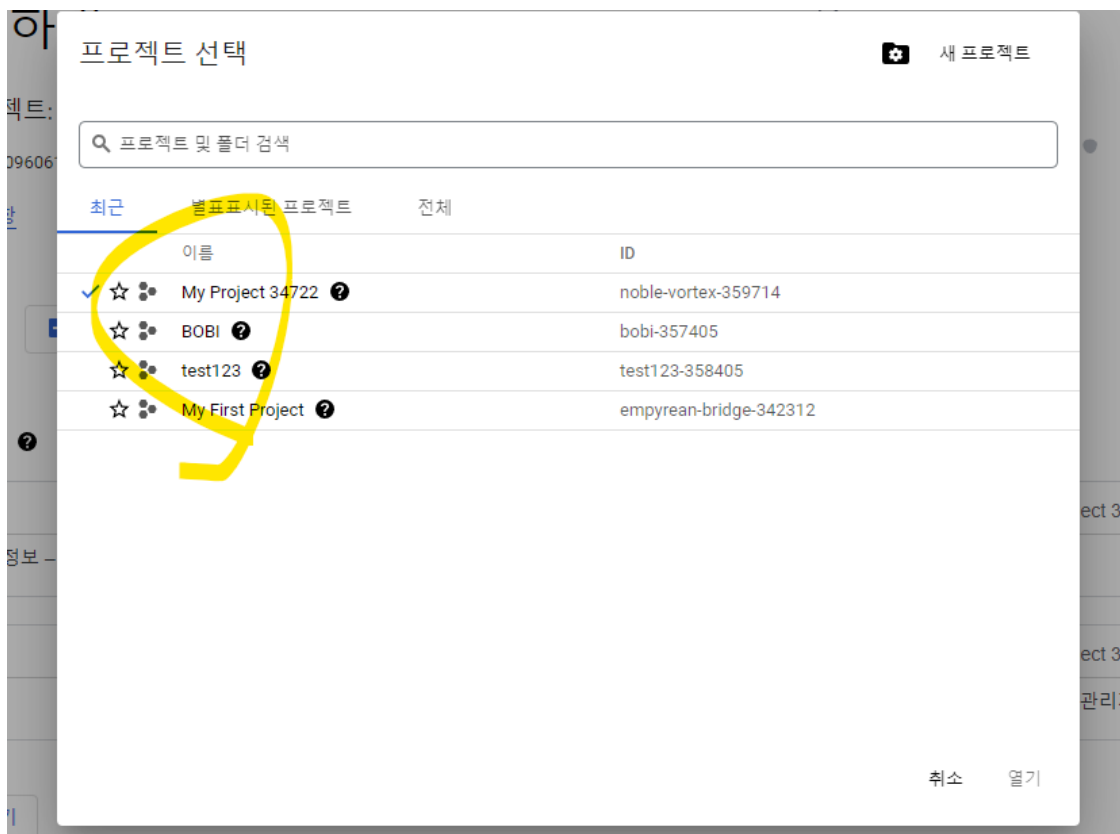
  s3.getObject(params, (err, data) => {
    if (err) {
      console.log(err, err.stack);
    }
    const blob = new Blob([data.Body], {
      type: "X-wav"
    })
    const blobURL = URL.createObjectURL(blob)
    setSource(blobURL)
  });
}, [])
```

5-2. Google Login (Client Side)

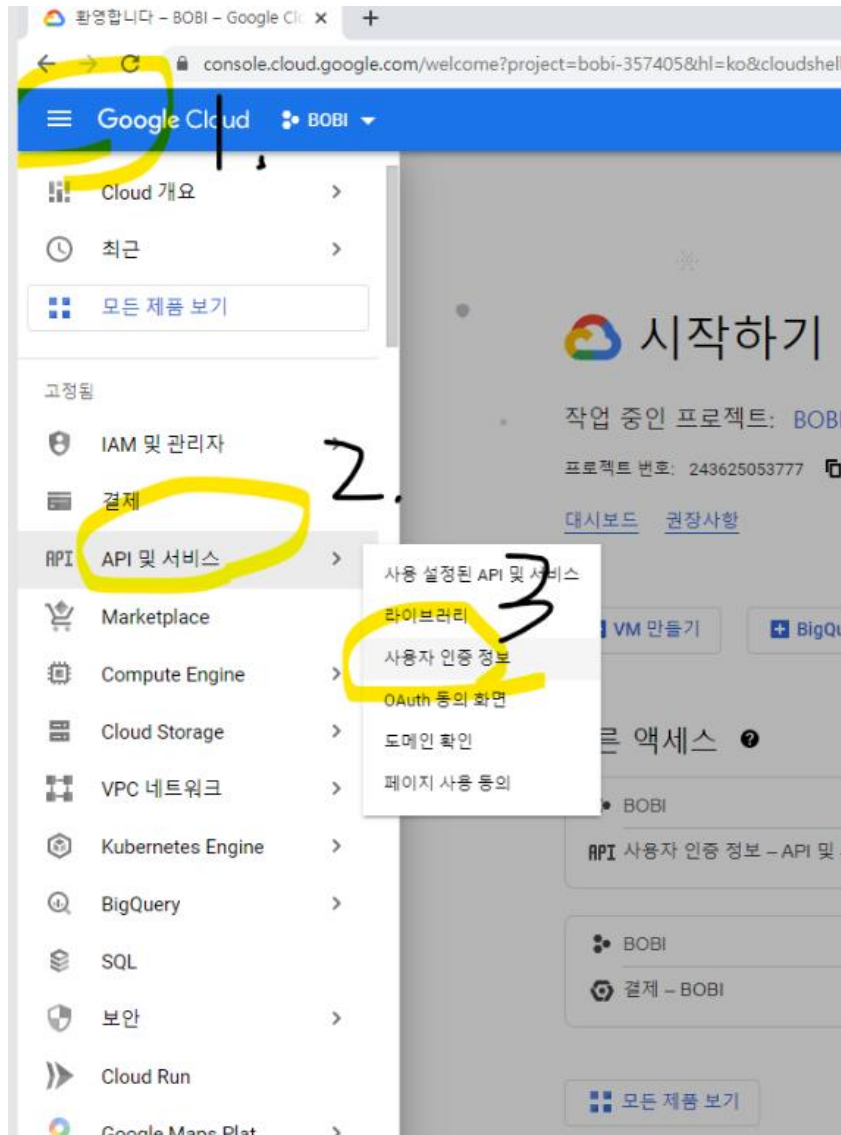
1. 구글 클라우드 플랫폼 접속해서 로그인하기 <https://console.cloud.google.com/>
2. 새 프로젝트 생성



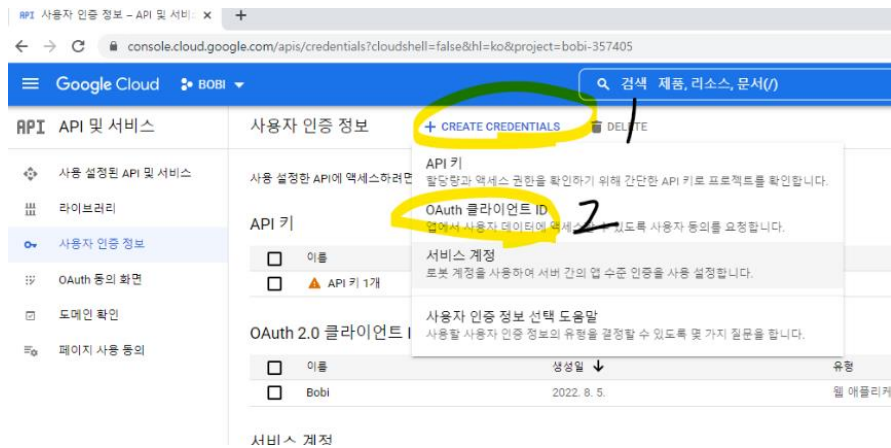
3. 프로젝트 탭 다시 클릭해서 생성한 프로젝트로 이동하기



4. 사용자 인증 정보로 이동



5. OAuth 클라이언트 ID 발급받기



6. 애플리케이션 유형 정하기

← OAuth 클라이언트 ID 만들기

클라이언트 ID는 Google OAuth 서버에서 단일 앱을 식별하는 데 사용됩니다. 앱이 여러 플랫폼에서 실행되는 경우 각각 자체 클라이언트 ID가 있어야 합니다. 자세한 내용은 [OAuth 2.0 설정](#)을 참조하세요. OAuth 클라이언트 유형을 [자세히 알아보세요](#).

애플리케이션 유형 *

- 웹 애플리케이션
- Android
- Chrome 앱
- iOS
- TV 및 입력 제한 기기
- 데스크톱 앱
- Universal Windows Platform(UWP)

7. 이름 입력하고 구글 로그인 적용할 사이트 URI 쓰고 리디렉션 URI 입력하기

Google Cloud BOBI

API API 및 서비스

← 웹 애플리케이션의 클라이언트 ID JSON 다운로드 보안

사용 설정된 API 및 서비스

라이브러리

사용자 인증 정보

OAuth 동의 화면

도메인 확인

페이지 사용 동의

이름 *

Bobi

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.

아래에 추가한 URI의 도메인이 승인된 도메인으로 OAuth 동의 화면에 자동으로 추가됩니다.

승인된 자바스크립트 원본

브라우저 요청에 사용

URI 1 *

https://17a208.p.ssafy.io

URI 2 *

http://localhost:3000

+ URI 추가

승인된 리디렉션 URI

웹 서버의 요청에 사용

URI 1 *

https://17a208.p.ssafy.io/

URI 2 *

http://localhost:3000/

URI 3 *

https://17a208.p.ssafy.io/login/

URI 4 *

http://localhost:3000/login/

8. 웹에 적용하기

```
import { GoogleLogin } from "react-google-login";
import { gapi } from "gapi-script";

...

const googleClientId = process.env.REACT_APP_GOOGLE_API_KEY;

useEffect(() => {
  function start() {
    gapi.client.init({
      clientId: googleClientId,
      scope: "email",
    });
  }
  gapi.load("client:auth2", start);
}, []);

const onSuccess = (response) => {
  // 로그인 성공했을 때 실행할 함수
};

const onFailure = (response) => {
  // 로그인 실패했을 때 실행할 함수
};

<GoogleLogin
  clientId={googleClientId}
  onSuccess={onSuccess}
  onFailure={onFailure}
/>
```

[주의] react-google-login은 npm install시 오류 발생 가능성 -- 타 라이브러리 대체 권장

6. .gitignore 파일 정보

- FE

bobi_frontend/.env

```
- REACT_APP_GOOGLE_API_KEY=243625053777-  
  t2htd7u0v85i9fnp0oq0cts7a3ba8tld.apps.googleusercontent.com  
- REACT_APP_S3_BUCKET=bobivoicebucket  
- REACT_APP_S3_REGION=ap-northeast-2  
- REACT_APP_S3_ACCESS_KEY=AKIAYF3ZGX73YPWSJUH3  
- REACT_APP_S3_SECRET_ACCESS_KEY=BirM+hx2UNkyYCXStqEtWAhHu9Bm5y3  
  nr1Y54l67  
-
```

- BE

bobi_backup/my_settings.py

```
- DATABASES = {  
-     'default': {  
-         'ENGINE': 'django.db.backends.mysql', #1 사용할 엔진  
-         (수정 X)  
-         'NAME': 'bobi', #2 연동할 MySQL 의 DB 이름  
-         'USER': 'pjt_bobi', #3 DB 접속 계정명  
-         'PASSWORD': 'mysql989312bobi#', #4 DB 접속 계정  
-         비밀번호  
-         'HOST': 'i7a208.p.ssafy.io', #5 실제 DB 주소 (따로 설정  
-         안했으면 수정 X)  
-         'PORT': '3306', #6 포트번호 (따로 설정 안했으면 수정 X)  
-     }  
- }  
- SECRET_KEY = 'django-insecure-  
  p^xm%mnC+betjp4@)lmx!35rbl^02kygw(@$u5(_bp3b$z_9a_'
```

7. File Structure

- FE

```

bobi_frontend
├── build
├── node_modules           // 설치 완료된 모듈
├── public
│   ├── index.html
│   └── manifest.json
├── src
│   ├── App.css
│   ├── App.jsx
│   ├── index.css
│   └── index.js
├── components
│   ├── ArchiveDropdown.js      // 사용 X
│   ├── ControlButton.jsx
│   ├── ControlButtonDummy.jsx  // 배포용 작동 X 버튼
│   ├── Dropdown.js
│   ├── GoogleButton.js        // 구글 로그인 버튼
│   ├── Graph.jsx              // 온도 그래프
│   ├── Graph1.jsx             // 습도 그래프
│   └── UserForm.jsx
│   └── archive
│       ├── ArchiveForm.jsx
│       ├── ArchiveImageForm.jsx
│       ├── ArchiveImageItem.jsx
│       ├── archiveImageWrite.jsx
│       ├── ArchiveVideoForm.jsx
│       └── ArchiveVideoItem.jsx
│   └── modal
│       ├── Modal.js
│       └── VoiceModal.js
│   └── story
│       ├── HiddenStory.jsx      // 해금 X 스토리 (클릭 X)
│       ├── StoryItem1.jsx
│       ├── StoryItem2.jsx
│       ├── StoryItem3.jsx
│       ├── StoryItem4.jsx
│       └── StoryItem5.jsx
└── voice
    └── VoicePlay.jsx
```

```
| VoiceRecord.jsx
|
|└─layout
|  | Layout.jsx          // 전체 레이아웃 적용
|  |
|  |└─footer
|  |  Footer.jsx
|  |
|  |└─header
|  |  ArchiveMenu.jsx
|  |  Header.jsx
|  |  HeaderMenu.jsx
|  |
|└─pages
|  | Config.jsx          // 환경 설정
|  | Control.jsx         // 로봇 조작
|  | Friendliness.jsx    // 친밀도
|  | Intro.jsx           // 서비스 소개
|  | Live.jsx            // 실시간 영상 확인
|  | Login.jsx           // 로그인
|  | Main.jsx            // 메인 페이지
|  | Sensor.jsx          // 센서 값 확인
|  | Story.jsx           // 해금된 스토리
|  | User.jsx
|  | UserDetail.jsx
|  | UserDetailEdit.jsx
|  | Voice.jsx           // 음성 송수신
|  |
|  |└─archive
|  |  ArchiveImage.jsx
|  |  ArchiveImageDetail.jsx
|  |  ArchiveImageUpdate.jsx
|  |  ArchiveVideo.jsx
|  |  ArchiveVideoDetail.jsx
|  |  ArchiveVideoUpdate.jsx
|  |  ArchiveVideoWrite.jsx
|  |
|└─package.json          // 설치 패키지 정보
```

- BE

```
BoBi
├──.config      // 서버 배포용 설정 파일
│   ├──nginx
│   └──uwsgi
├──accounts      // user 정보
├──archives      // 영상 아카이브
├──bobi          // 센서, 친밀도 등 로봇 저장 데이터
├──bobi_backend  // url, DB 세팅 등
├──bobi_frontend // FE 빌드 파일
├──movements
├──stories
├──voices
└──my_settings.py // DB 설정 등 (개인정보 포함)
```



감사합니다

2022 SSAFY 공통 프로젝트 2반 8팀

CARE ROBOT BoBi

장명근 곽다원 김동원 신선영 이승훈 정재훈