



# 한 번 시도로 안되면

두 번 시도하는 백엔드 개발자 손준혁입니다.



손준혁

📞 010-3524-9445

✉️ sonjuhy@gmail.com

🎓 창원대학교 컴퓨터공학과 ( 2015.03 - 2021.08 )

💻 삼성 청년 SW 아카데미 7기

🔗 [Github](#) 📖 [기술 블로그](#)

## About Me

- 궁금한 것에 대해 공부하고 탐구하는 걸 두려워하지 않습니다.
- 계획을 세우면 실천과 결과 산출까지, 끝까지 추진합니다.
- 실생활에서 불편하거나 개선하고 싶은 걸 직접 코딩으로 해결해보고자 합니다.
- 구현한 것에 대해 만족하지 않고 개선 및 기능 추가를 하는 꾸준한 발전을 합니다.

## Skill set( )

### ▼ 원하는 기능 구현 가능

- |                   |               |
|-------------------|---------------|
| ◆ JAVA            | ◆ Spring Boot |
| ◆ ubuntu          | ◆ nginx       |
| ◆ Spring Data JPA | ◆ mysql       |
| ◆ MQTT(Mosquitto) | ◆ Git         |
| ◆ Android(Java)   |               |

### ▼ 사용&응용 해본 적 있음

- |           |                        |
|-----------|------------------------|
| ◆ WebRTC  | ◆ Socket I/O           |
| ◆ Kafka   | ◆ Spring Cloud Gateway |
| ◆ Docker  | ◆ Spring Security      |
| ◆ Jenkins | ◆ React.js             |
| ◆ MongoDB | ◆ JWT                  |
| ◆ Eureka  |                        |

▼ 문서참고 가능

- ◆ python
- ◆ JUnit
- ◆ React Native
- ◆ Vue.js
- ◆ Yolo V5

▼ 개발도구

- Eclipse
- IntelliJ
- Android Studio
- Visual Studio Code
- Mysql Work Bench
- PyCharm
- MobaXterm

## Awards

이름	성과	주관	일자	프로젝트
SSAFY 공통 프로젝트	우수상	삼성 청년 SW 아카데미	2022.08	<u>Octop-Us</u>
SSAFY 특화 프로젝트	우수상	삼성 청년 SW 아카데미	2022.10	<u>탐정: 렌즈 속 비밀</u>
SSAFY 자율 프로젝트	우수상	삼성 청년 SW 아카데미	2022.11	<u>가게 사장</u>
SSAFY 자율 프로젝트	자율 프로젝트 결선 발표회 입상	삼성 청년 SW 아카데미	2022.11	<u>가게 사장</u>

## Projects

### 프로젝트 리스트

Aa 이름	:☰ 태그
<u>MyHome</u>	개인 프로젝트
<u>Octop-Us</u>	SSAFY
<u>탐정: 렌즈 속 비밀</u>	SSAFY
<u>가게 사장</u>	SSAFY



# MyHome Project 2.0

## MyHome Project Ver 2.0

웹, 앱 어플리케이션

2023.01 ~

개인 프로젝트

### 담당 직무

- Front-End
- Back-End
- Server
- CI/CD
- etc...

### 담당 업무

- Project Structure Design
  - Back-End
    - Spring Boot

- Django
- Front-End
  - NEXT JS
- Server
  - Ubuntu
- Web (NEXT JS)
  - Sign In &Up
    - Back-end [JAVA] : JWT, Spring Security
    - Front-End : REST(POST), storage
  - Title Page
    - Check Access Rights
  - Main Page
    - Main
      - Components : Notice, Weather, Light (IoT)
    - Notice
      - Append
      - Delete
      - Inquiry
    - Weather
      - Change Location
      - Inquiry detail information
    - Light (IoT)
      - Remote control of the light
  - Cloud
    - Upload : Multiple Uploads, preview about image
    - Download : Multiple Downloads
    - Make new Folder
- Application (Android)
  - Update Android SDK Version : 29 → 32
  - Apply MVVM Pattern
  - Change Weather API Server (Direct API → Spring Boot API)
  - Change DB Communicatation route (PHP → Spring Boot)
  - Change Cloude Service Access route (SSH → Spring Boot)
  - Change the Authentication method for user info (user info → JWT)

- Refactoring Code about HTTP feature (AsyncTask → Retrofit)
- Back-End (Spring Boot, Django)
  - Spring Boot
    - Sign In & Up
    - Check Access Right
    - Weather API
      - Get Weather info from the Meteorological Administration API and Reformat data
    - Notice
    - IoT (Light)
      - Recive : Http
      - Send : Kafka
    - Cloud
      - Upload (Multiple, preview about image)
      - Download (Multiple)
      - Video Stream
      - File Delete
      - Check All Files Status in Cloud Server
        - Get Signal : Kafka
  - Django
    - Check Light Status Periodically and Save info to DB
      - Comm : MQTT
      - Cycle : 1min
    - Light Control
      - Send & Recive : MQTT
    - Send Signal (Check files) to Spring Boot
- ETC
  - DB (Maria DB)
    - Modify Table, Column based on Normalization
    - Modify Table, Column name based on Naming Rules
  - Voice recognitaion based light control
    - Use Google STT service
  - ESP8266
    - Change Button Processing to Interrupt

## 성과

---

- Improved Security And Reliability
- Improved Scalability
- Additional features increase Accessibility and Convenience

## 링크

---

- Spring Boot
  - link : <https://github.com/sonjuhy/MyHomeSpring>
- Django
  - link : <https://github.com/sonjuhy/MyHomeDjango>
- NEXT JS
  - link : <https://github.com/sonjuhy/MyHomeNextJS>
- ESP 8266
  - link : <https://github.com/sonjuhy/MyHomESP8266>
- Android (Regacy Code : No updates due to security related to server IP)
  - link : [https://github.com/sonjuhy/MyHome\\_Open](https://github.com/sonjuhy/MyHome_Open)

## 개발 환경 & 기술 스택

---

- IDE
  - IntelliJ
  - Visual Studio Code
  - PyCharm
  - Android Studio
  - Arduino
  - MobaXterm
- Tec
  - Spring Boot
  - Spring Security
  - JWT
  - JPA
  - Django
  - Kafka (+ zookeeper)

- Mosquitto (MQTT)

- MariaDB

- NEXT JS

- Docker

- Jenkins

- Language

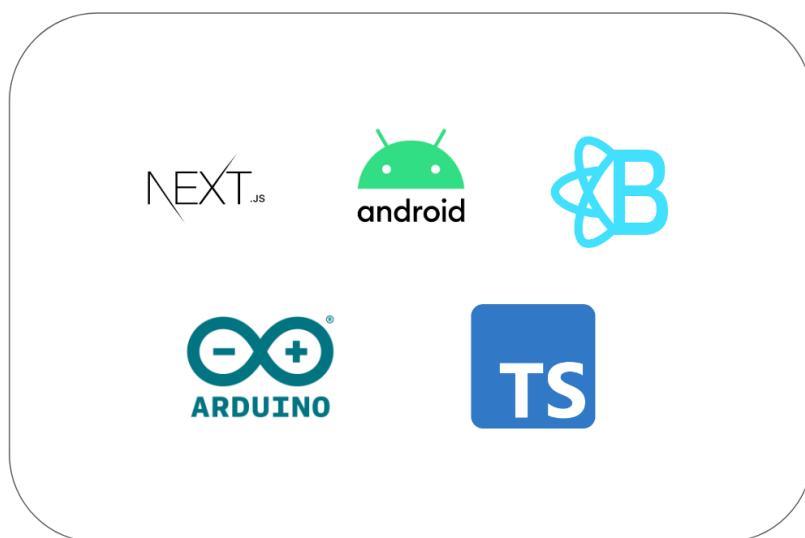
- JAVA

- Python

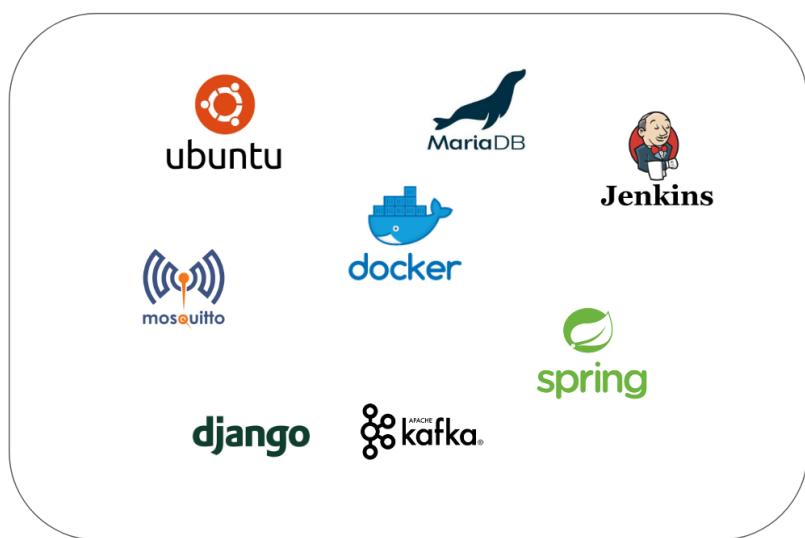
- TypeScript

- C++

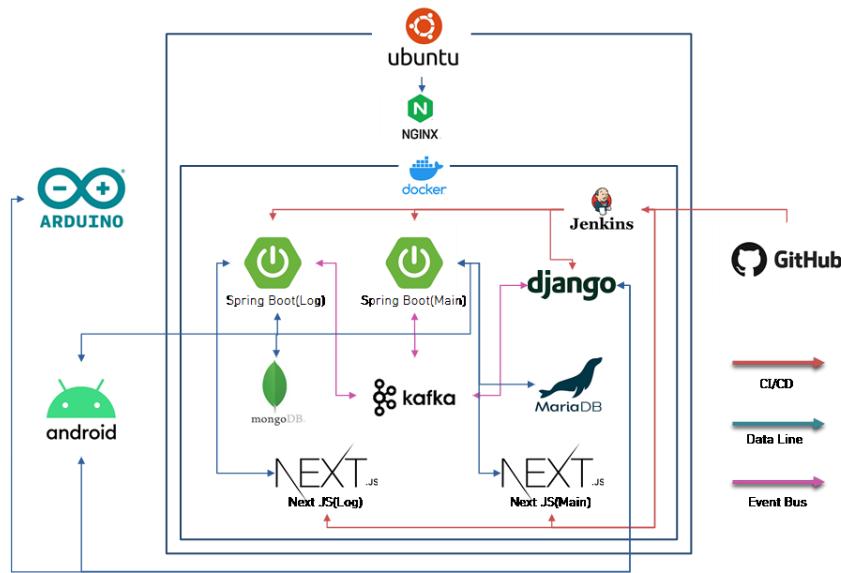
## Client



## Back-End



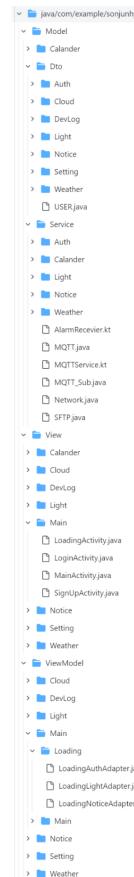
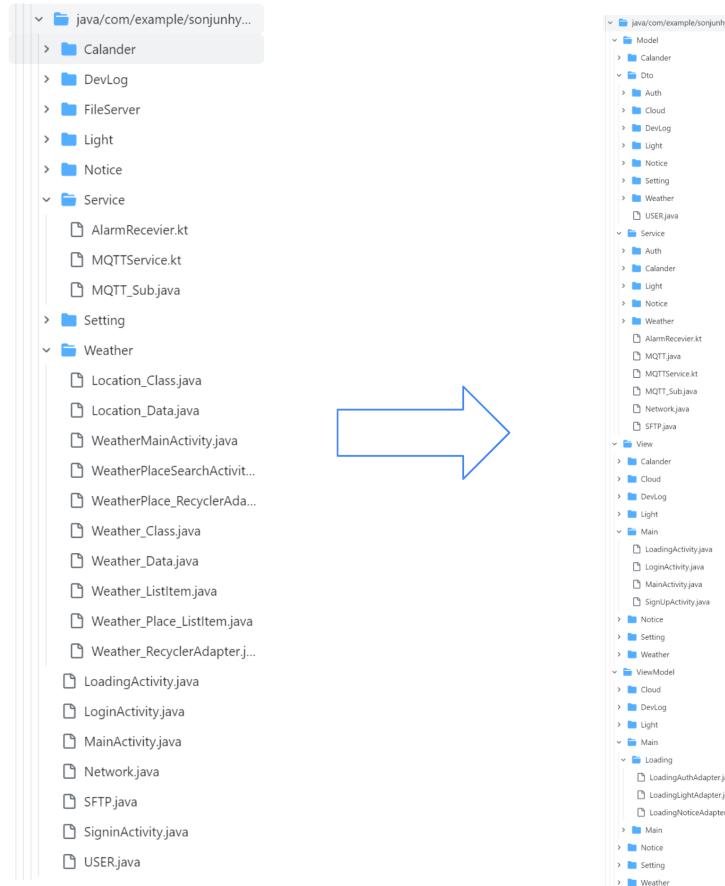
## 구조도



## 안드로이드 코드 리팩토링

### MVVM 패턴 적용

No Pattern → MVVM Pattern



## Deprecated 코드 변경

### AsyncTask → Retrofit2

```
public class Network extends AsyncTask<String, Void, String> {
    ...
    @Override
    protected void onPreExecute() {
        if(dialog_use) {
            progressDialog = new ProgressDialog(context);
            progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            progressDialog.setCancelable(false);
            progressDialog.setMessage("로딩 중입니다.");
            progressDialog.show();
        }
        super.onPreExecute();
    }

    @Override
    protected String doInBackground(String... _param) {
        String n = _param[0];
        switch (n){
            case "login":
                link += "/Login_Check.php";
                Login_Input(_param[1], _param[2]);
                mode = 1;
                upload_mode = true;
                break;
            case "Signin":
                link += "/Signup.php";
                Signin_Input(_param[1], _param[2], _param[3]);
                mode = 2;
                upload_mode = true;
                break;
            case "IDOverlap":
                link += "/SignUp_IDCheck.php";
                Overlap_Input(_param[1]);
                upload_mode = true;
                break;
            case "Get_Userinfo":
                link += "/Get_UserInfo.php";
                mode = 3;
                break;
        }
    }
}

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.Path;

public interface AuthService {
    @GET("/auth/getUserInfo/{accessToken}")
    Call<UserDto> getUserInfo(@Path("accessToken") String token);

    @POST("/auth/signIn")
    Call<String> signIn(@Body UserDto dto);

    @POST("/auth/signUp")
    Call<String> signUp(@Body UserDto dto);
}

public class LoadingAuthAdapter {
    private final static String baseURL = "http://192.168.0.18:8080";
    private int MINDACTIVITY_CODE = 100;
    private int LOADINGACTIVITY_CODE = 200;

    private AuthService service;

    private Retrofit retrofit;

    public LoadingAuthAdapter(){
        retrofit = new Retrofit.Builder()
            .baseUrl(baseURL)
            .addConverterFactory(ScalarsConverterFactory.create())
            .addConverterFactory(JsonConverterFactory.create())
            .build();
        service = retrofit.create(AuthService.class);
    }

    public void login(Context loadingActivityContext, SharedPreferences sp, String id, String pw, boolean weatherLoading, ProgressDialog progressDialog) {
        UserDto tmpDto = new UserDto(id, pw);
        System.out.println("LoadingAuthAdapter AutoLoading tmpDto : " + tmpDto);
        callString callUserDto = service.signIn(tmpDto);
        callUserDto.enqueue(new Callback<String>() {
            ...
        });
    }

    public void onResponse(Call<String> call, Response<String> response) {
        System.out.println("Login onResponse isSuccessfull : " + response.isSuccessful());
        if(response.isSuccessful()){
            System.out.println("Login onResponse body : " + response.body());
        }
    }
}
```



# 서비스 사진

## WEB

### Main Page

Only Access User

The screenshot shows the main page of the MyHome application. On the left is a dark sidebar with navigation links: 정회원님 (My Account), 설정 (Settings), Home (selected), Light, Weather, and Cloud. The main content area has a header with the logo and 'about'. Below it is a 'Notice' section with a title 'Notice Title' and a note: 'This is a space that displays what you have written as a notice.' It also shows the last update by 'admin'. The 'Weather' section displays a large image of a road through a desert landscape under a cloudy sky, with text indicating '맑음' (clear), wind speed '2.3m/s', temperature '23 °C', and precipitation '강수량: 강수없음'. The 'Light Control' section shows seven light bulb icons, each with a label: 작은 회장실, 안방 뒷 스위치, 안방 아래 스위치, 부엌 싱크대, 부엌 탁자, 거실장, and 거실 뒷 스위치.

This screenshot shows the same main page as above, but the 'Light Control' section is more detailed. It includes two rows of light controls. The top row contains seven controls: 작은 회장실 (ON), 안방 뒷 스위치 (OFF), 안방 아래 스위치 (ON), 부엌 싱크대 (ON), 부엌 탁자 (ON), 거실장 (ON), and 거실 뒷 스위치 (ON). The bottom row contains five controls: 거실 중간 스위치 (ON), 거실 아래 스위치 (ON), 중간방 뒷 스위치 (ON), 중간방 아래 스위치 (ON), and 작은방 스위치 (OFF).

## Notice Page

### Notice Main Page

The screenshot shows the Notice Main Page interface. On the left is a dark sidebar with navigation links: 정회원님 (Profile), 설정 (Settings), Home, Light, Weather, and Cloud. The main area has three sections:

- TOP Notice:** A box titled "TOP Notice" containing a "Notice Title" section with placeholder text "This is a space that displays what you have written as a notice." and a "작성자" (Author) field showing "admin".
- ADD Notice:** A box titled "ADD Notice" with fields for "제목" (Title) and "내용" (Content), both with placeholder text "제목을 입력하세요." and "내용을 입력하세요.". A blue "생성" (Create) button is at the bottom.
- Notice List:** A table with columns "Notice Title", "작성자" (Author), and "작성일" (Created Date). It lists one item: "title\_3" by "admin" on "2023년 5월 25일".

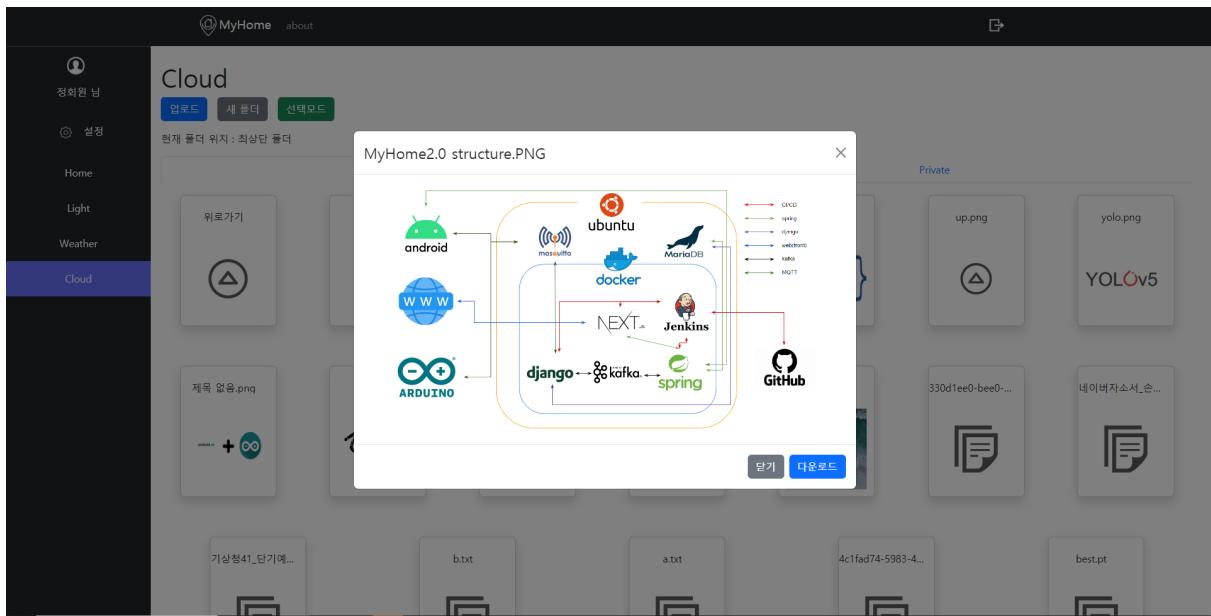
## Cloud Page

### Cloud Main Page

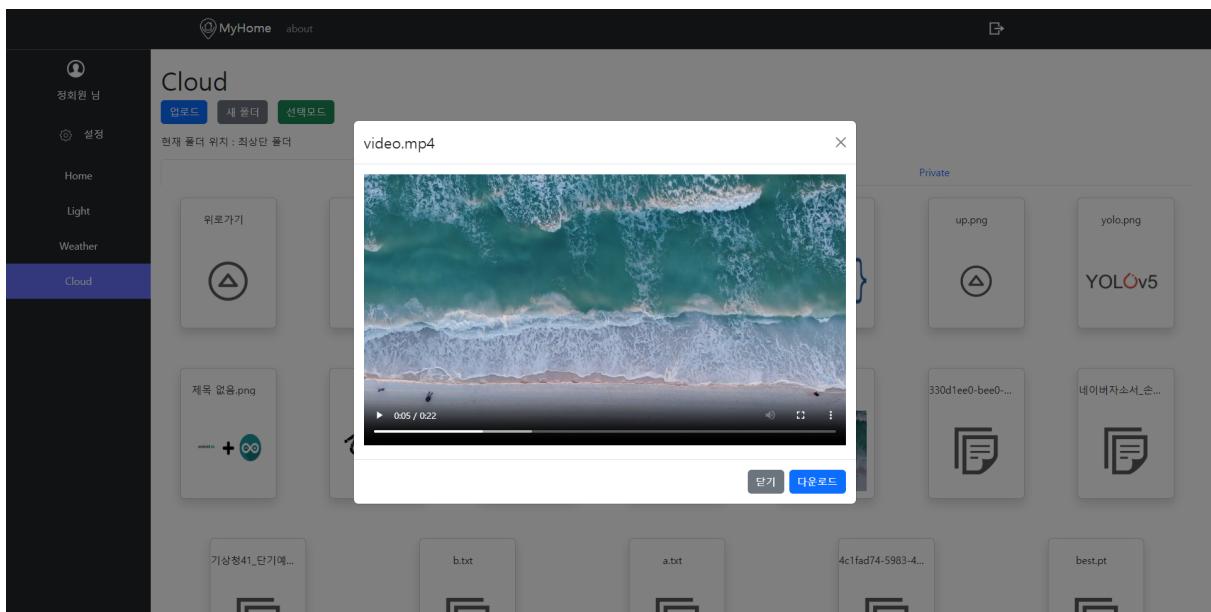
The screenshot shows the Cloud Main Page interface. On the left is a dark sidebar with navigation links: 정회원님 (Profile), 설정 (Settings), Home, Light, Weather, and Cloud (which is selected and highlighted in blue). The main area shows a grid of files categorized into "Public" and "Private".

Category	File Name	Description
Public	위로가기	Up arrow icon
	test2	Folder icon
	python.png	Python logo icon
	db.png	Database icon
	json.png	JSON icon
	up.png	Up arrow icon
	yolo.png	YOLOv5 icon
	YOLov5	YOLOv5 icon
	제작 없음.png	Placeholder icon
	11.png	Image icon
Private	MyHome2.0 stru...	Diagram icon
	personal_logo.P...	Logo icon
	video.mp4	Video icon
	330d1ee0-bee0-4...	File icon
	네이버소서_순...	Document icon
	best.pt	File icon
	기상청41_단기예...	Weather icon
	b.txt	Text file icon
a.txt	Text file icon	
4c1fad74-5983-4...	File icon	

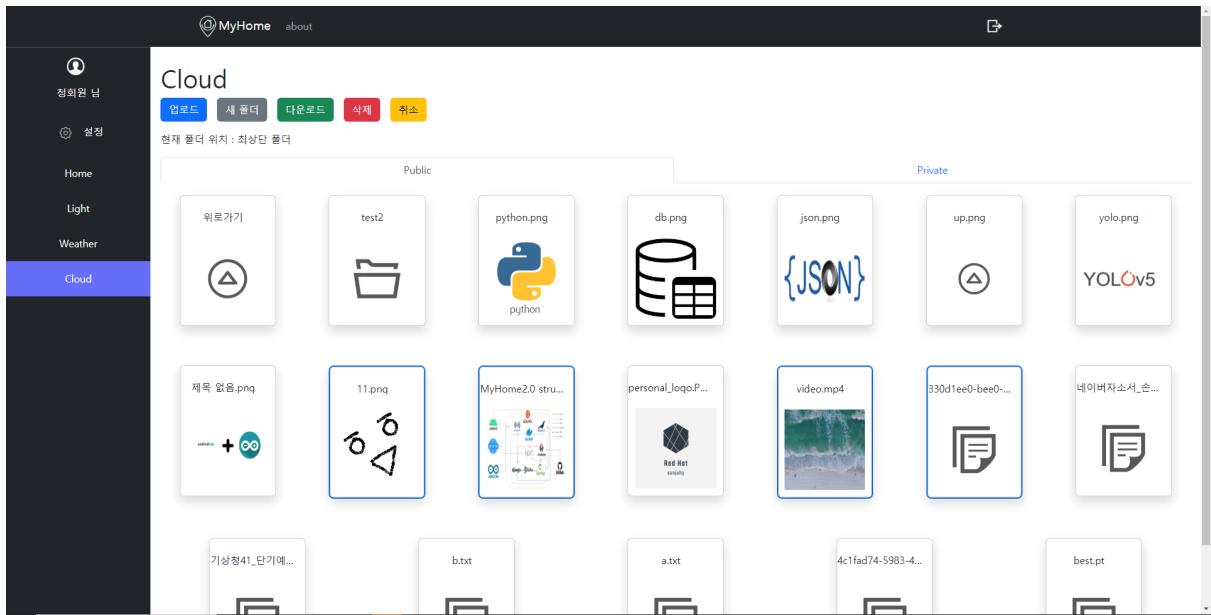
## Cloud Main Page [Image View]



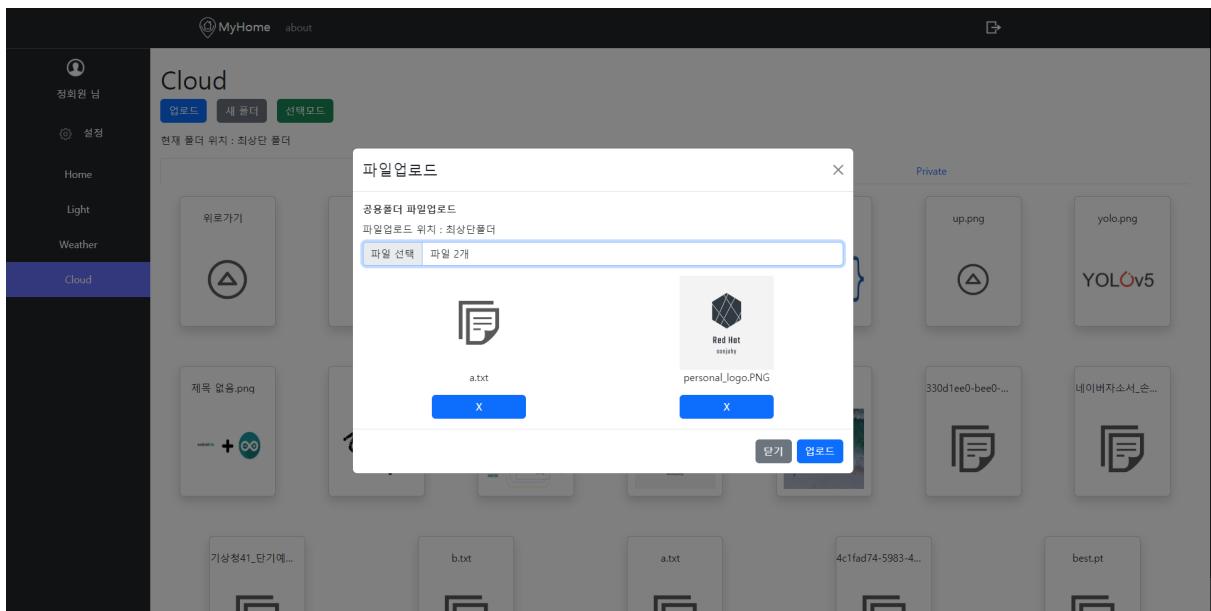
## Cloud Main Page [Video Streaming]



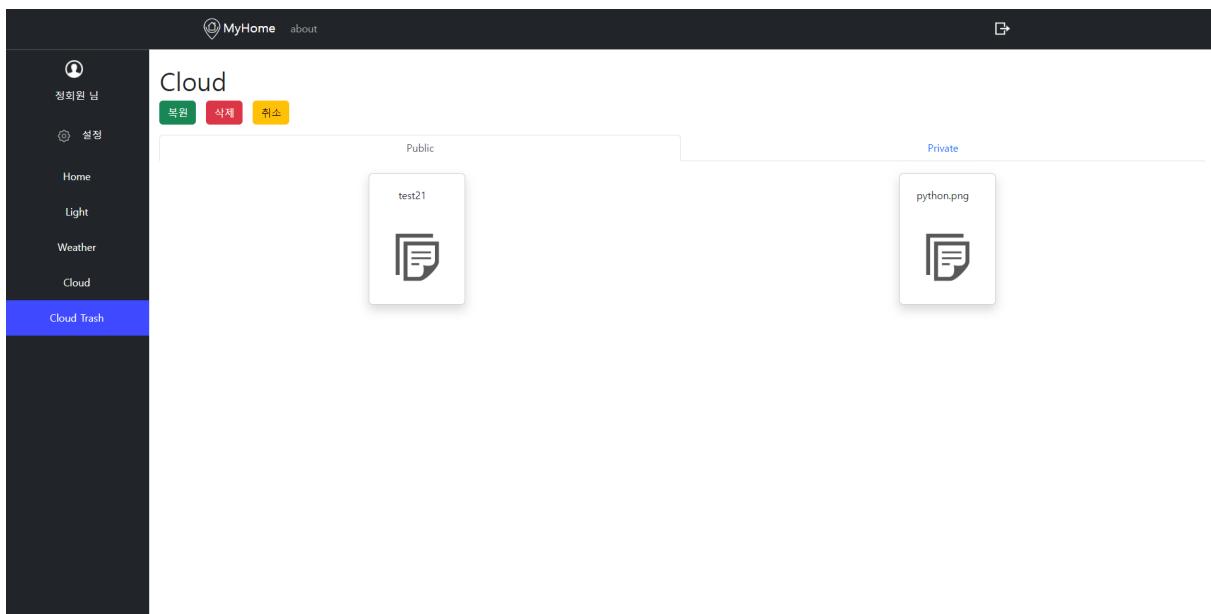
## Cloud Main Page [Multiple Select & Download, Delete]



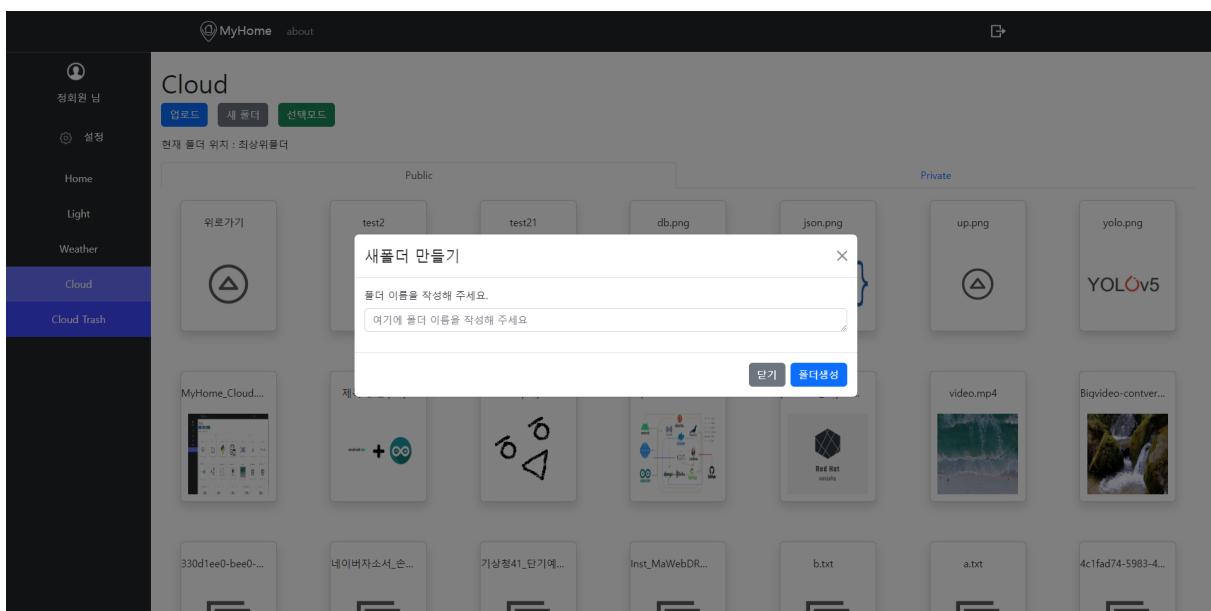
## Cloud Main Page [Multiple Upload, Preview about Image]



## Cloud Main Page [Move to Trash Folder, Restore File in Trash Folder]



## Cloud Main Page [Make New Folder]





# MyHome Project

## 개요

20년도 더 된 집에서 원격으로 전등을 ON/OFF 컨트롤하고 예약까지 한다면 어떨까 그리고 추후 지원하는 기기를 늘려 최대한 많은 부분을 원격으로 상태 확인, 컨트롤 하길 원해 시작한 프로젝트

## 구상

- 원격 클라이언트 : 안드로이드
  - 가족 구성원 모두 갤럭시 스마트폰을 사용하므로 한번 개발하면 모두 사용 가능한 안드로이드 플랫폼을 선택
  - IoT 컨트롤 뿐만 아니라 파일 서버, 날씨, 일정 공유 또한 한 어플내에 가능하도록 올인원 플랫폼으로 제작하고자 함
- 서버 : 우분투
  - 공부하는 김에 OS 설치부터 서버 설정까지 다 해보기 위해 무료 OS이면서 국내 포럼이 큰 편인 Ubuntu 18.04 LTS 버전 선택
  - 가족 구성원만 사용 가능한 파일 서버를 만들어 사진, 영상 등 정보를 최대 10TB 까지 지원
- iot : 전등 스위치
  - 도시가스 밸브, 방범문 도어락과 같이 안전과 직결된 부분보다 안전 문제에 대해 비교적 안전하고 가장 사용성이 높은 전등 스위치부터 도전

## 기능 요구사항

기능명	카테고리	작동 결과	설명
회원가입	회원	회원가입 신청	서비스 사용을 위한 회원가입
로그인	회원	유저 정보 리턴	가입된 회원 정보 가져오기
자동 로그인	회원	자동 로그인 켜기	자동로그인 기능 활성화
로그아웃	회원	로컬 유저 정보 삭제	현재 기기에 저장된 유저 정보 삭제
메인 화면 날씨	날씨	저장된 위치 날씨 정보제공	메인 화면에 날씨 정보 간소화 제공
상세 날씨	날씨	저장된 위치 날씨 정보제공	날씨 세부 정보 제공
날씨 위치 저장	날씨	표시 원하는 위치 저장	원하는 날씨 위치 정보 저장
메인 화면 날씨 정보 표시	날씨	메인 화면 날씨 표시 ON/OFF	메인화면에 보이는 날씨 정보 ON/OFF
메인화면 전등 컨트롤	IoT(전등)	메인 화면에서 전등 컨트롤	메인 화면에서 전등 컨트롤할 패널
메인화면 전등 즐겨찾기	IoT(전등)	메인 화면 전등 즐겨찾기	메인 화면에서 즐겨찾기 선택한 전등 따로 표시
전등 세부 목록, 컨트롤	IoT(전등)	방 별로 전등 세부 선택 및 컨트 롤	방 별로 전등 별도 세부 컨트롤 및 예 설정 진입
전등 즐겨찾기 등록, 해제	IoT(전등)	전등 즐겨찾기	전등 즐겨찾기 등록, 해제
전등 작동 예약	IoT(전등)	전등 작동 예약 등록	전등 작동을 예약으로 등록
전등 예약 수정	IoT(전등)	저장된 전등 예약 정보 수정	저장된 전등 예약 정보 수정
전등 예약 목록 관리	IoT(전등)	저장된 전등 예약 정보 관리	저장된 전등 예약 리스트 삭제, 수정
전등 상황 실시간 캡신	IoT(전등)	전등 ON/OFF 실시간 캡신	어플 실행 중 전등 상태 변동값 실시간 캡신 및 반영
파일 서버 개인 공간 탐색	파일 서버	서버 내 개인 공간 탐색	서버 내 개인 공간 탐색 및 검색
개인 파일 서버 업&다운로 드	파일 서버	서버 내 개인 공간에서 파일 업 로드 및 다운로드	서버 내 개인 공간에서 파일 업로드, 다 운로드 및 수정
파일 서버 공용 공간 탐색	파일 서버	서버 내 공용 공간 탐색	서버 내 공용 공간 탐색 및 검색
공용 파일 서버 업&다운로 드	파일 서버	서버 내 공용 공간에서 파일 업 로드 및 다운로드	서버 내 공용공간에서 파일 업로드, 다 운로드 및 수정
파일 임시삭제(휴지통)	파일 서버	파일 서버 내 파일 임시 삭제	파일 서버 내 파일 휴지통 이동
임시 파일 복구	파일 서버	임시삭제 파일 위치 복구	휴지통 내 파일 위치 복구
파일 영구 삭제	파일 서버	임시삭제 파일 영구 삭제	휴지통 내 파일 영구 삭제
공지사항 등록	공지	공지사항 등록	공지사항 등록
공지사항 삭제	공지	공지사항 삭제	공지사항 삭제
공지사항 메인화면 표시	공지	공지사항 메인화면 캡신	공지사항 메인화면에 캡신
어플리케이션 업데이트	시스템	어플 자체 업데이트 기능	어플 자체 업데이트 기능 구현

## 개발 환경

- Android Studio
- PyCharm
- Arduino IDE

## 사용 라이브러리

### ▼ Switch(ESP8266)

- Http
- MQTT Client

### ▼ Android

- MQTT Client
- REST

### ▼ Server(Python)

- MQTT Server
- MQTT Client Python
- Schedule

### ▼ Server(DevOps)

- Nginx
- PHP
- MariaDB

## 사용 기술스택

- JAVA
- Python
- PHP
- C++

## 서비스 화면

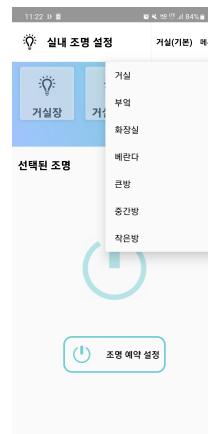
### • 메인 화면



### • 파일 서버



### • 조명 컨트롤



### • 조명 예약목록



## 서비스 로그

- DB(MariaDB) 스위치 컨트롤로그

2022-11-28	7:2	kitchen table	Off	self
2022-11-28	7:8	Living Room3	On	self
2022-11-28	7:10	Living Room3	Off	self
2022-11-28	7:40	kitchen table	On	self
2022-11-28	7:45	big Room1	Off	self
2022-11-28	7:54	kitchen table	Off	self
2022-11-28	8:12	middle Room1	On	self
2022-11-28	8:29	middle Room1	Off	self
2022-11-28	10:10	bathRoom2	On	self
2022-11-28	10:11	bathRoom2	Off	self
2022-11-28	10:49	bathRoom2	On	self
2022-11-28	11:0	bathRoom2	Off	self
2022-11-28	12:19	bathRoom2	On	self
2022-11-28	12:24	bathRoom2	Off	self
2022-11-28	12:38	big Room1	On	self
2022-11-28	12:48	bathRoom2	On	self
2022-11-28	12:52	bathRoom2	Off	self
2022-11-28	12:58	kitchen table	On	self
2022-11-28	13:17	kitchen table	Off	self
2022-11-28	13:27	middle Room1	On	self
2022-11-28	14:0	middle Room1	Off	self
2022-11-28	14:38	bathRoom2	On	self
2022-11-28	14:43	bathRoom2	Off	self
2022-11-28	14:43	big Room1	Off	self
2022-11-28	14:51	big Room1	On	self
2022-11-28	14:54	big Room1	Off	self

19251 rows in set (0.01 sec)

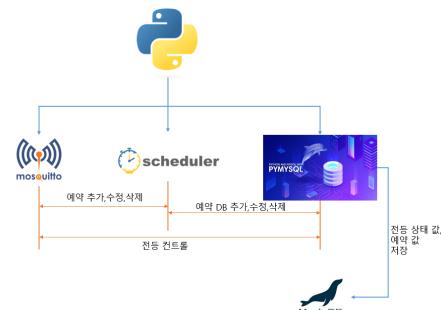
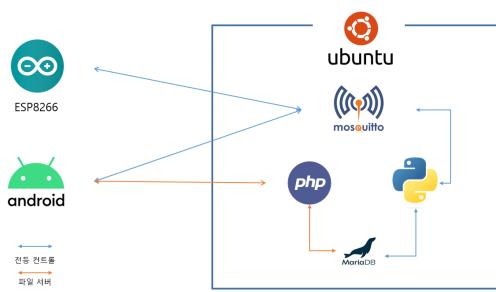
- DB(MariaDB) 스위치 작동 예약 리스트

Name	Time	Room	Do	Day	Activated	Reiteration	RoomKor	Num
꼴자	23:30	big Room1	OFF	월,화,수,목,금,	False	True	안방 윗 스위치	11
굿모닝~^	6:30	big Room1	ON	월,화,수,목,금,	False	True	안방 윗 스위치	12

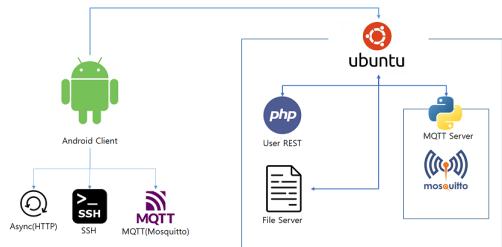
## 구조도

- 메인 구조

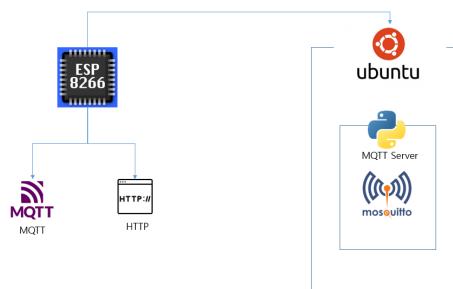
- 서버 (Python)



- 안드로이드



- 스위치



카페  
나 **한식집** 할 건데 어디에 차리지?  
중국집

보러 가기

# 가게 사장

태그

SSAFY

## 가게사장

웹 어플리케이션

17 2022.10 - 2022.011

삼성 청년 SW 아카데미 자율 프로젝트

## 참여 인원

- Back-End 3명
- Front-End 3명

## 담당 직무

Backend, Server, CI/CD

## 담당 업무

- 회원가입 & 로그인 [백엔드 (JAVA, JWT, Spring Security) ]
- MSA 설계
- Load Balancing (Spring Cloud Gateway, Eureka)
- Server 세팅 [DevOps]
  - Docker
  - Jenkins

- Kafka
- SSL
- Nginx
- Mysql
- MongoDB
- Redis

## 성과

---

2022.11 SSAFY 자율 프로젝트 우수상 (1등)

2022.11 SSAFY 자율 프로젝트 결선발표회 입상

## 링크

---

- 개발 [Notion](#) 바로가기

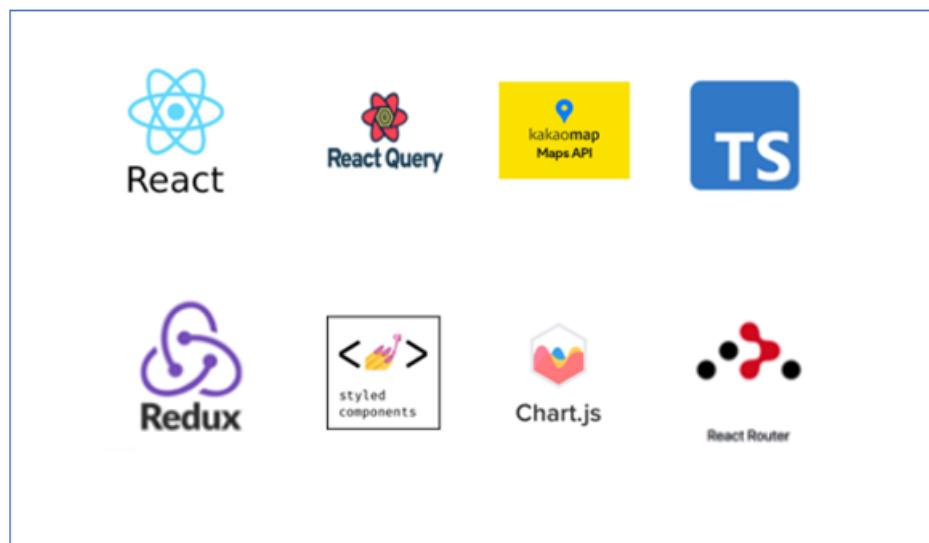
## 개발환경 & 기술스택

---

- IDE
  - IntelliJ
  - VSC
  - MobaXterm
- Lib
  - Spring Boot
  - Spring Cloud
  - Eureka
  - JWT
  - JPA
  - Swagger
  - Spring Security
  - Kafka
  - React
  - Flask
  - Redis
  - MongoDB
  - Mysql
  - Jenkins

- Docker
- language
  - JAVA
  - Python
  - Java Script(Type script)

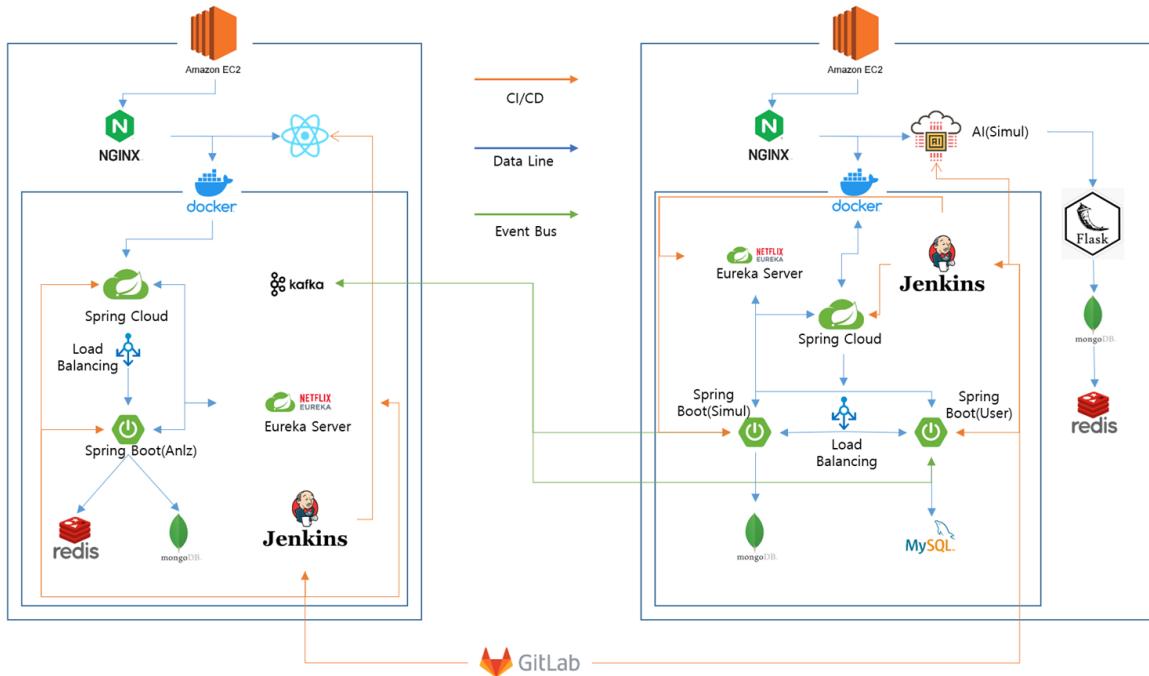
## Front-End



## Back-End



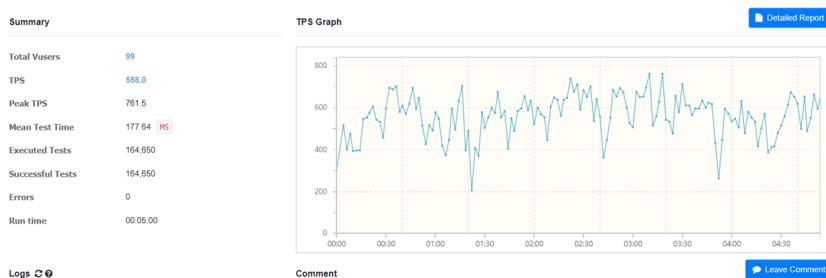
## 구조도



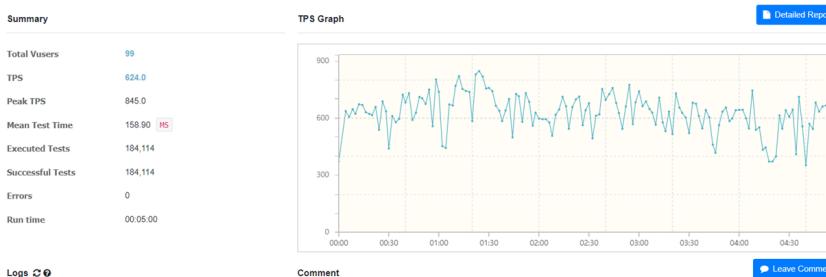
## 서버 성능

- 로드 밸런싱

- 사용목적 : 서비스에 오는 트래픽을 분산하여 안정적인 서비스 운영 및 성능 향상
- 테스트 툴 : nGinder
- LB OFF



- LB ON



- 결과

- 전반적인 성능 11~12% 향상

- DB

- 동시성

- MySQL에서 select for update 기능을 사용
- 동시성 제어를 위하여 특정 데이터(ROW)에 대해 베타적 LOCK을 검
- 사용 예시

```
public interface UserRepository extends JpaRepository<UserEntity>
{
    @usage 🔒 손준혁
    @Lock(LockModeType.PESSIMISTIC_FORCE_INCREMENT)
}
```

- 트랜잭션 롤백

- JPA에서 지원하는 기능을 사용, 데이터 처리 중 문제 발생 시 롤백
- 사용 예시

```
1 usage 🔒 손준혁
@Transactional
@Modifying(clearAutomatically = true)
@Query(value = "update user set state=:state where email=:email")
int saveByEmail(@Param("state")int state,
                @Param("email")String email);
```

- MSA

- 독립적 서비스 실행
- 서버 별 실행 컨테이너
  - 1번 서버

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
3a224914d220	back/anlz	"java -jar /app.jar"	11 days ago	Up 11 days	
67da19b08c7d	spring_anlz_3	"java -jar /app.jar"	11 days ago	Up 11 days	
baf450c4bc62	spring_anlz_2	"docker-entrypoint.s..."	11 days ago	Up 11 days	
ba450c4bc62	mongo	"docker-entrypoint.s..."	11 days ago	Up 11 days	
0a9edcc84830	mongodb-container	"java -jar /app.jar"	13 days ago	Up 13 days	
ef6072629c34	back/gateway	"java -jar /app.jar"	13 days ago	Up 13 days	
bff0e53d37b1	redis	"docker-entrypoint.s..."	2 weeks ago	Up 2 weeks	
cd8adce783d0	redis-container	"java -jar /app.jar"	2 weeks ago	Up 2 weeks	
spring_eureka	spring_eureka	"java -jar /app.jar"	2 weeks ago	Up 2 weeks	

- 2번 서버

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ee700bc06c69	back/simul	"java -jar /app.jar"	11 days ago	Up 11 days	
0a3e82597d0c	back/simul	"java -jar /app.jar"	11 days ago	Up 11 days	
b45fbfb766ec	back/user	"java -jar /app.jar"	13 days ago	Up 13 days	
3f9d8bf93637	back/user	"java -jar /app.jar"	13 days ago	Up 13 days	
4d3f59ac9d05	back/gateway	"java -jar /app.jar"	2 weeks ago	Up 2 weeks	
cd8adce783d0	redis	"docker-entrypoint.s..."	2 weeks ago	Up 2 weeks	
r					
cabccbcb03968	back/eureka	"java -jar /app.jar"	3 weeks ago	Up 3 weeks	
e495340943a6	mongo	"docker-entrypoint.s..."	3 weeks ago	Up 3 weeks	
ner					

- 각 서비스 별 DB 독립
- 서비스 별 데이터 전송은 Kafka 사용
  - 비동기 통신으로 각 서비스간 결합도를 낮춤

## 서비스 화면

### 회원가입 & 로그인 페이지

#### 회원가입 기능

- 회원가입을 할 수 있습니다.

당신의 비즈니스 파트너, 가게사장

ID(E-mail)  
gagessajang@email.com

PASSWORD  
수자, 영어, 특수문자 포함 9~15자

PASSWORD CONFIRM  
위에 입력한 비밀번호와 동일하게 입력

NICKNAME  
한글 2~10자, 영어 3~15자, 특수문자 불포함

#### 로그인 기능

- 로그인을 할 수 있습니다.

당신의 비즈니스 파트너, 가게사장

ID  
gagessajang@email.com

PASSWORD  
수자, 영어, 특수문자 포함 9~15자

로그인 유지

## 메인 페이지

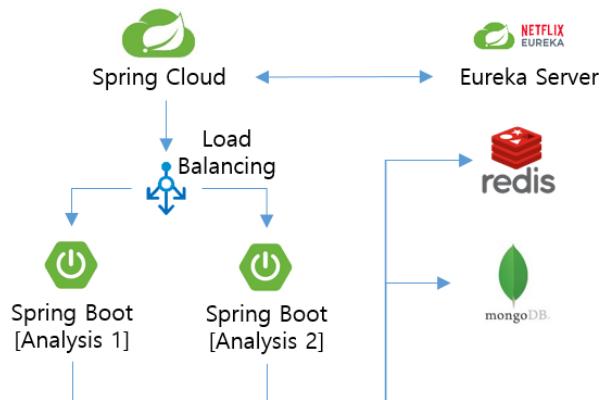
### 업종 분석 기능

- 선택한 업종에 대한 분석을 해줍니다.



## 프론트 ↔ 백 커뮤니케이션

- Eureka & Gateway 를 이용한 Swagger
  - 구조
    - 구성
      - Eureka Sever : 1개
      - Spring Cloud Gateway : 1개
      - Spring Boot : 2개
    - 구조도



- Eureka

- 각 서비스 포트 관리

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
APIGATEWAY-SUB-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">apigateway-sub-service:455736a5067e342fe051671a43dda36a</a>
BACKEND-SIMUL	n/a (2)	(2)	UP (2) - <a href="#">ip-172-26-12-6.ap-northeast-2.compute.internal:5d06df75b6b58240cb4e0d6efa6675d5:0</a> , <a href="#">ip-172-26-12-6.ap-northeast-2.compute.internal:b6690ad08814b7fb69d1c21cfac0f5:1:0</a>
BACKEND-USER	n/a (2)	(2)	UP (2) - <a href="#">ip-172-26-12-7.ap-northeast-2.compute.internal:36a473a07fec73e951484c9418ec3aa1:0</a> , <a href="#">ip-172-26-12-6.ap-northeast-2.compute.internal:56ccde6cc137a9ecd4e1f44e1de1a42:0</a>

- Gateway

- 각 서비스 별 Path, Filter 설정

```
spring:
  application:
    name: apigateway-service
  cloud:
    gateway:
      routes:
        - id: openapi
          uri: lb://${spring.application.name}
          predicates:
            - Path=/v3/api-docs/**
          filters:
            - RewritePath=/v3/api-docs/(?<path>.*), /${path}/v3/api-docs
        - id: backend/anlz-service
          uri: lb://BACKEND_ANLZ
          predicates:
            - Path=/backend/anlz/**, /anlz/**
          filters:
            - RewritePath=/backend/anlz/(?<path>.*), /${path}
            - CustomFilter
            - name: LoggingFilter
              args:
                baseMessage: Spring Cloud Gateway Logging Filter
                preLogger: true
                postLogger: true
```

- Swagger

- 한 도메인에서 여러개의 컨테이너로 분리된 서비스를 통합 조회 가능

Simul 서비스 Swagger

The screenshot shows the Swagger UI interface for a 'User 서비스 Swagger'. At the top, there's a navigation bar with the 'Swagger' logo and a dropdown menu labeled 'Select a definition' set to 'backend/user'. Below the header, the title 'Practice Swagger' is displayed with a '1.0 OAS3' badge. A sub-header indicates the path '/v3/api-docs/backend/user' and a note 'practice swagger config'. In the main content area, under the heading 'auth-controller Auth Controller', there are four API endpoints listed:

- GET /user/auth/getUserInfo** getUserInfoOnlyNickname
- GET /user/auth/hello** test hello
- GET /user/auth/kakao** kakaoLogin
- GET /user/auth/kakao/expiration** kakaoLogOut

A 'Servers' dropdown at the top left is set to 'http://k7e2051.p.ssafy.io:8081 - Inferred Url'. There are also collapse/expand arrows for each endpoint entry.

User 서비스 Swagger



## Octop-Us

태그

SSAFY

### 문어 마피아 게임 'Octop-US'

웹 어플리케이션

2022.07 - 2022.08

삼성 청년 SW 아카데미 공통 프로젝트

#### 담당 직무

Backend, Frontend, Server

#### 담당 업무

- 회원가입 & 로그인 [백엔드]
- OpenVidu(WebRTC) Server 구현
- OpenVidu Client 구현 [프론트엔드]
- 미니게임(낚시꾼 게임) [백&프론트엔드]
- Server 세팅(Docker, Nginx, DB, SSL) [DevOps]

#### 참여 인원

- Back-End 3명
- Front-End 3명

# 성과

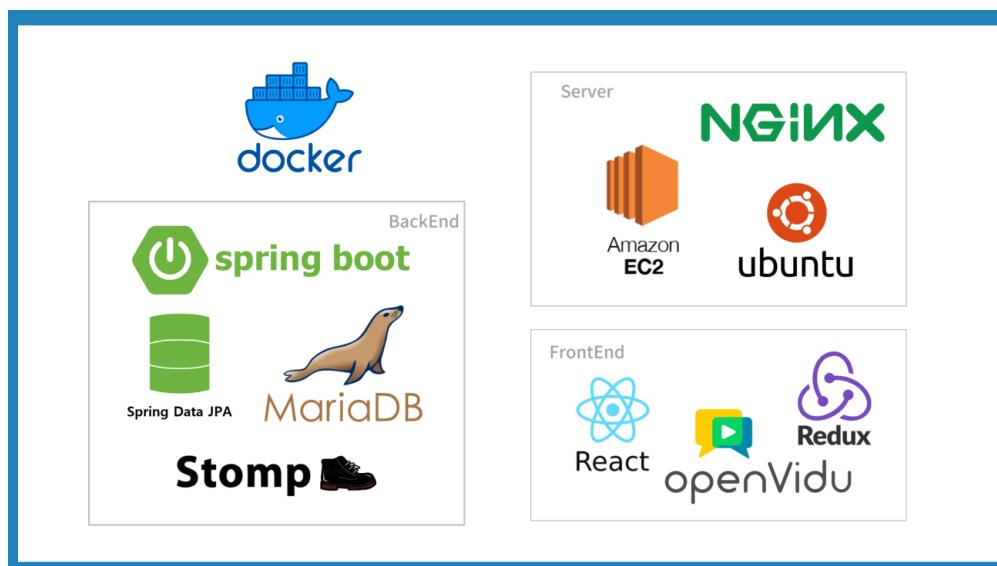
2022.09 SSAFY 공통 프로젝트 우수상 (2등)

## 링크

- Github 바로가기
- 개발로그 [Notion](#) 바로가기

## 개발환경 & 기술스택

- IDE
  - IntelliJ
  - VSC
  - MobaXterm
- Lib
  - Spring Boot
  - JPA
  - Swagger
  - React
  - OpenVidu(WebRTC)
- language
  - JAVA
  - Java Script
- 기술스택



# 서비스 화면

## WebRTC(OpenVidu) 사용한 화상채팅

### 인게임 화상채팅

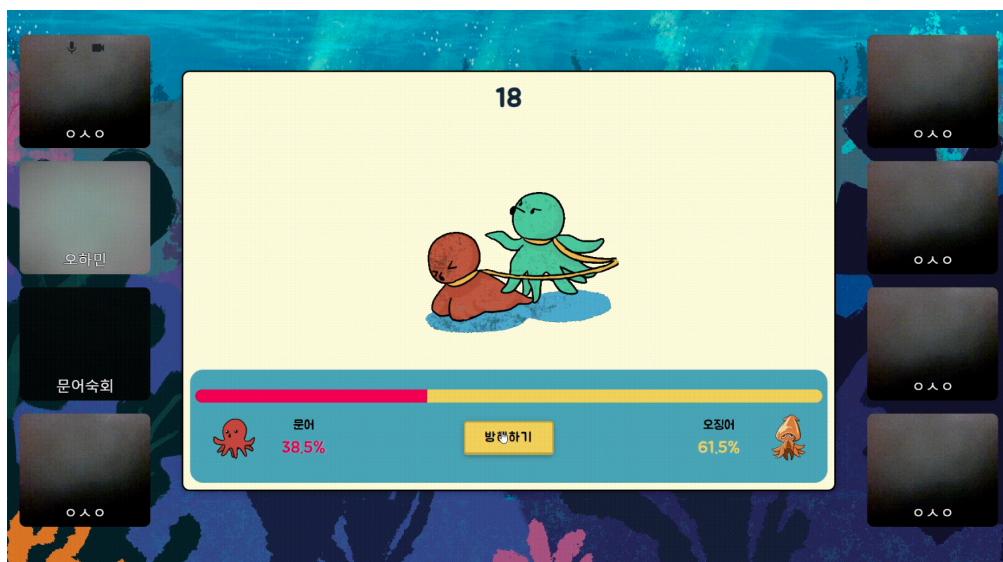
- 각자 캠으로 인게임에서 WebRTC를 이용한 화상 채팅



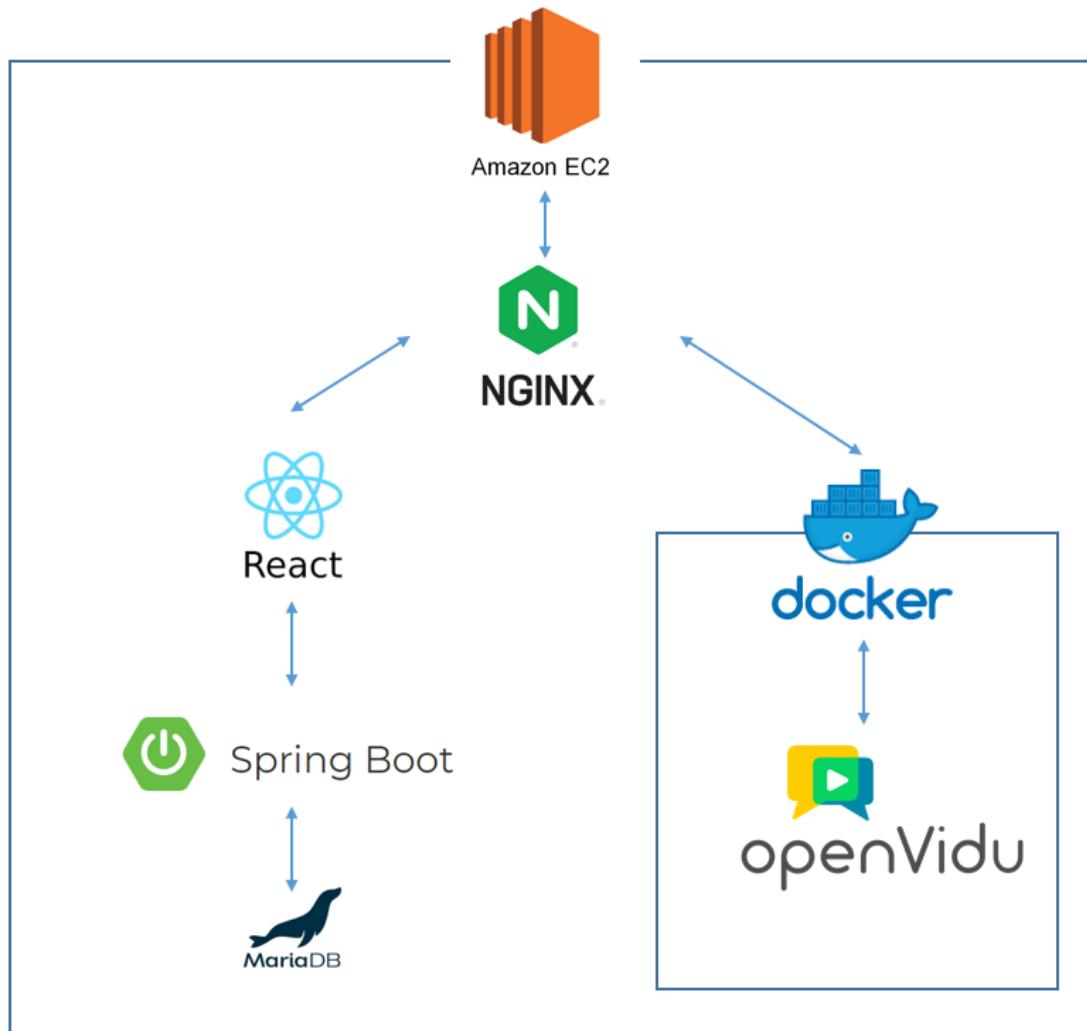
## 미니게임 기능

### 낚시꾼 미니게임

- 오징어 vs 문어 팀으로 클릭횟수의 평균값에 따른 승패를 결정합니다.



## 구조도



## OpenVidu Server 구현

- 사전 설치
  - Docker
- 설치
  1. 다운로드

```
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

2. conf 파일 설정
  - a. /opt/openvidu 폴더에 .env 파일 열기
  - b. ssl 인증서 관련 설정을 아래 사진과 같이 수정

```

# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=198.51.100.1

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=MY_SECRET

# Certificate type:
# - selfsigned: Self signed certificate. Not recommended for production use.
# - owncert: Users will see an ERROR when connected to web page.
# - letsencrypt: Valid certificate purchased in a Internet services company.
#   Please put the certificates files inside folder ./owncert
#   with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#   required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#   variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=songjuhy@gmail.com

```

DOMAIN IP, OPENVIDU\_SECRET, TYPE, EMAIL 설정

### c. 포트 수정

```

HTTP_PORT=80
HTTP_PORT=80

```

### 3. CertBot 설치

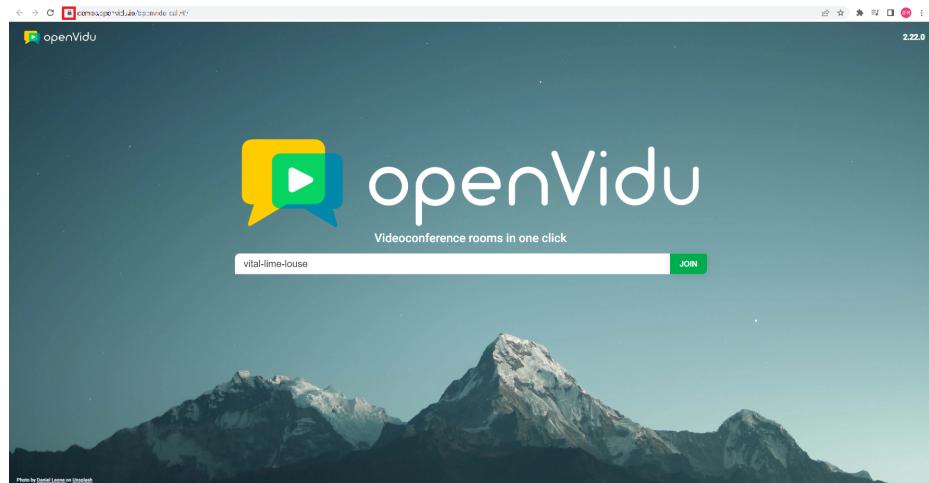
```
sudo apt-get install letsencrypt -y
```

### 4. 실행

- a. OpenVidu 설치된 폴더로 이동
- b. 아래 명령어로 실행

```
./openvidu start
```

### c. 해당 포트로 사이트가 아래 사진처럼 잘 보이는지 확인



HTTPS로 연결되었는지 확인 (빨간 사각형 안에 자물쇠 모양)

### d. 확인 후 포트를 원하는 포트로 변경 후 재시작

- 추후 nginx를 따로 설치할 경우 반드시 변경



# 탐정: 렌즈 속 비밀

태그

SSAFY

## 탐정: 렌즈 속 비밀

모바일(안드로이드) 어플리케이션

2022.08 - 2022.010

삼성 청년 SW 아카데미 특화 프로젝트

### 담당 직무

AI(Yolo V5), Frontend(React, React Native - Camera), Server

### 담당 업무

- 데이터 전처리 [AI]
- 데이터 학습 [AI]
- 이미지 디텍팅 [AI]
- Android Application 배포용 사이트 (템플릿 사용) [프론트엔드]
- Android Application 빌드 (React Native)
- Server 세팅(Ai, Nginx, DB, SSL) [DevOps]

## 참여 인원

---

- Back-End 1명
- Front-End 2명
- 디자인 1명
- 스토리 1명
- AI 1명

## 성과

---

2022.10 SSAFY 특화 프로젝트 우수상 (1등)

## 링크

---

- Github 바로가기
- 개발로그 Notion [바로가기](#)

## 개발환경

---

- IDE
  - PyCharm
  - IntelliJ
  - VSC
  - MobaXterm
  - Blender
- Lib
  - Spring Boot
  - React
  - React Native
  - Yolo V5
  - PyTorch
  - mysql
- language
  - JAVA
  - Python
  - Java Script

# AI [Image Detecting]

## Yolo V5 설정

- 다운로드
  - 원하는 위치에 아래 명령어로 소스 다운로드

```
git clone https://github.com/ultralytics/yolov5.git
```

- 라이브러리 설치
  - 아래 명령어로 라이브러리 설치

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

- 데이터 위치 설정 (train.py)
  - 기존 데이터 경로 및 정보파일 이름 수정

```
parser.add_argument('--data', type=str, default=ROOT / 'data/sample.yaml', help='dataset.yaml path')
```

- worker 수 조정
  - worker 수 조정
- 데이터 정보 설정 (.yaml)
  - yolo 폴더 내 data로 이동 후 yaml 파일 생성
  - 아래 내용 추가 및 기존 데이터 값 추가

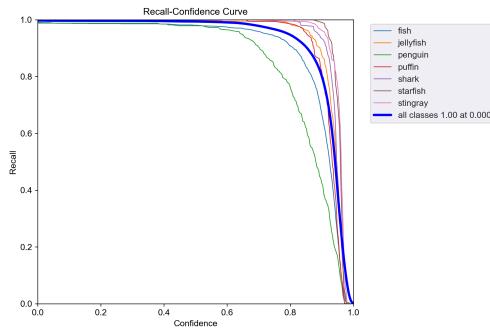
```
path: ../../datasets/sampleData# dataset root dir
train: train/images# train images (relative to 'path') 128 images
val: train/images# val images (relative to 'path') 128 images
test: # test images (optional)
nc: 7
names: ['fish', 'jellyfish', 'penguin', 'puffin', 'shark', 'starfish', 'stingray']
```

- path : 이미지, 라벨 폴더가 존재하는 루트 폴더
- train : 학습시킬 이미지가 있는 폴더
- nc : 이미지에 있는 클래스 갯수
- names : 이미지에 있는 클래스 이름 (앞에서부터 0번)

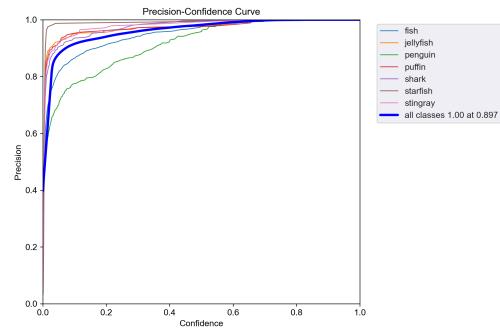
## Train

- 데이터 전처리 - 라벨 파일 양식 : 클래스 명, 네 모서리 좌표값(4개)
- train.py 파일 실행
- 결과

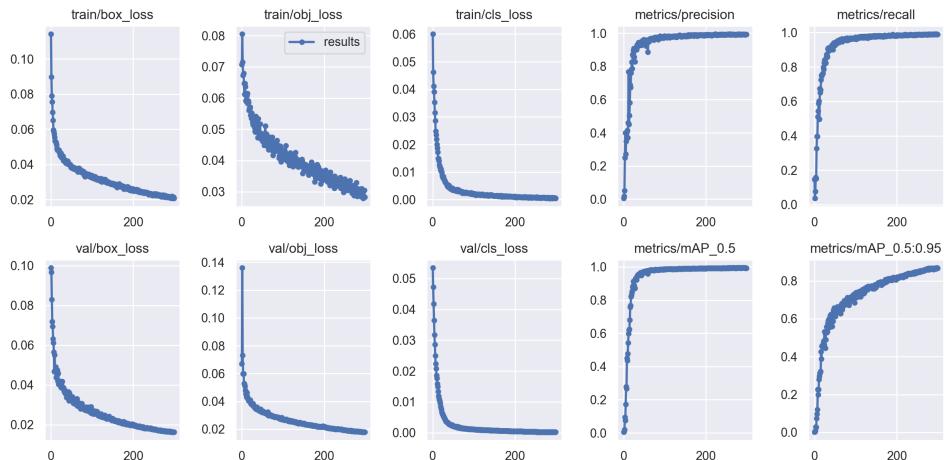
- R\_curve



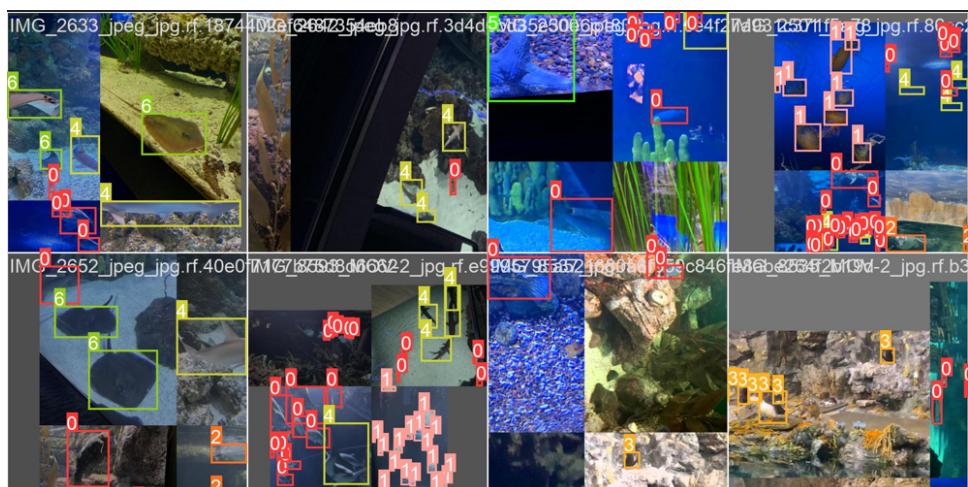
- P\_curve



- results



- batch

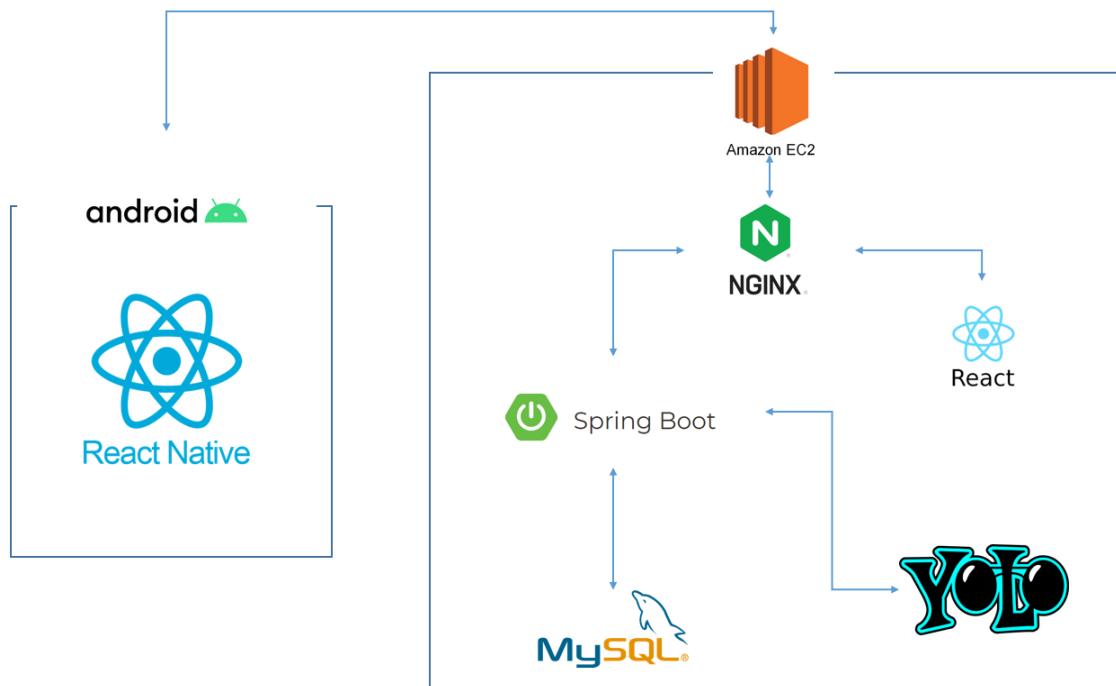


## Detect

- detect.py 실행



## 구조도

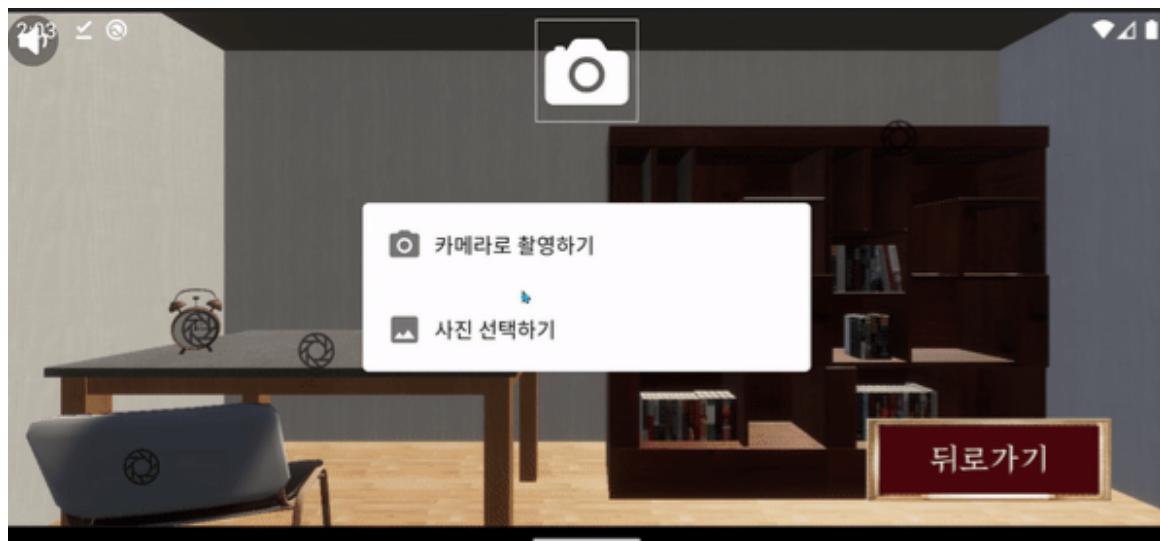


## 서비스 화면

### 사진 선택

#### 사진 선택(혹은 촬영)

- 사진을 선택(혹은 촬영) 후 서버로 전송하여 분석 결과를 요청합니다.



### 사진 분석 결과 (정답)

#### 사진 분석 결과 (정답)

- 전송한 사진의 분석 결과가 해당하는 단서가 있을 경우 스크립트가 진행됩니다.



## 사진 분석 결과 (오답)

### 사진 분석 결과 (오답)

- 전송한 사진의 분석 결과가 해당하는 단서가 없을 경우 아래 사진처럼 진행됩니다.



## 사진 분석 결과 (중복)

### 사진 분석 결과 (중복)

- 전송한 사진의 분석 결과가 이미 조사한 단서일 경우 아래 사진처럼 진행됩니다.

