

# 포팅매뉴얼



## 목차

1. 개발 환경
2. 계정 정보
3. 프로퍼티 파일 목록
4. 빌드 및 배포
  - a. EC2 환경설정
  - b. Nginx 설정
  - c. DB 설정
  - d. CI/CD 구축
  - e. 로컬 실행 가이드
5. 외부 서비스
  - a. OpenVidu
  - b. 카카오지도
  - c. Teachable Machine

## 개발환경

### Frontend

- **Node** - npm 10.2.5
- **Framework** - React 18.2.0
- **Build tool** - Vite 5.0.8
- **State management** - jotai 2.6.2
- **Language** - TypeScript 5.2.2

- **HTTP client** - axios 1.6.5
- **Style** - Styled Components 6.1.8

## Backend

- **Java - 17** Oracle OpenJDK 17.0.9
- **Framework** - Spring Boot 3.2.1
- **ORM** - JPA(Hibernate)
- **Dependency tool** - gradle 8.5

## Server

- Ubuntu 20.04 LTS
- Nginx 1.18.0 (Ubuntu)
- Docker 25.0.0
- OpenVidu 2.29.0
- AWS S3

## Database

- MySQL 8.0.34
- Redis 7.2.4

## UI/UX

- Figma

## IDE

- IntelliJ IDEA 2023.2.5 (Ultimate Edition)
- WebStorm 2023.3.2

# 계정 정보

## mySQL

- HOST : [i10c111.p.ssafy.io:3306](http://i10c111.p.ssafy.io:3306)
- ID : pawsitive
- PW : gkrkddPd12~!

## Redis

- HOST : [i10c111.p.ssafy.io:6379](http://i10c111.p.ssafy.io:6379)
- Database: 3
- PW : gkrkddPd12!@

## sonarqube

- ID : admin
- PW : gkrkddPd12!@

## Jenkins

- HOST : <http://i10c111.p.ssafy.io:8080>
- ID : pawsitive
- PW : gkrkddPd12!@

## Openvidu

- URL : <https://i10c111.p.ssafy.io:8443>
- PW: PAWSITIVE

## Ubuntu

```
ssh -i {pemkey위치}\I10C111T.pem ubuntu@i10c111.p.ssafy.io
```

- public ip: 3.36.63.3
- private ip: 172.26.8.148

## 프로퍼티 파일 목록

- 위치

```
backend
├── src
│   ├── main
│   │   ├── resources
│   │   │   └── application.properties
```

- application.properties - 프로젝트, 서버, DB, Swagger, AWS S3, SMTP, Openvidu 설정 정보

## 빌드 및 배포

### EC2 환경설정

- 사용 포트

Port	내용
22	SSH
80	HTTP
443	HTTPS
3306	MySQL
3478	TURN/STUN
6379	Redis

Port	내용
8080	Jenkins
8081	SpringBoot
8090	React
9000	SonarQube

## Nginx 설정

- nginx 설치

```
$ sudo apt-get install
$ nginx nginx -v
```

- SSL 인증서 발급

```
$ sudo apt-get install letsencrypt
$ sudo systemctl stop nginx
$ sudo letsencrypt certonly --standalone -d

# 설치 잘 되었는지 확인
$ cd /etc/letsencrypt/live
$ ls
# SSL 인증서 발급 시 입력한 도메인과 같은 명의 폴더가 있는지 확인
```

- nginx 설정

```
$ cd /etc/nginx/conf.d/
$ vim default.conf
```

### default.conf

```
# 443 포트로 접근시 ssl을 적용한 뒤 포트로 요청을 전달해주도록 하는 설정
server {
    server_name i10c111.p.ssafy.io;
    client_max_body_size 100M;
```

```

        location / {
            proxy_pass http://127.0.0.1:8090;

            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }

        location /api {
            proxy_pass http://127.0.0.1:8081;
        }

        location ~^/(swagger-ui|v3) {
            proxy_pass http://127.0.0.1:8081;
        }

        location /ws/chat {
            proxy_pass http://127.0.0.1:8081;

            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }

        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/i10c111.p.;
        ssl_certificate_key /etc/letsencrypt/live/i10c111.;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; #

    }

    server {
        return 301 https://$host$request_uri;
    }

```

```
        listen 80;
        server_name i10c111.p.ssafy.io;
        return 404;

    }
```

## DB 설정

### MySQL

- 영구 보존 위해 명시 볼륨 생성

```
$ docker volume create mysql-volume
```

- 생성한 볼륨을 컨테이너에 마운팅 해 실행

```
$ docker run -d --name mysql-container -p 3306:3306 -v my:
```

- bash 셸로 접속

```
$ docker exec -it mysql-container bash
```

- mysql 서버 접속

```
$ mysql -u root -p # root 계정 id: root, pw: 1234
```

- 유저 생성 및 권한 부여

```
$ create user pawsitive identified by 'gkrkddPd112~!';
$ grant all privileges on *.* to pawsitive;
$ flush privileges;
$ exit;
```

```
$ docker run -d --name mysql-container -p 3306:3306 -v mys
```

## REDIS

```
$ docker exec -it redis-container redis-cli --raw
```

```
AUTHENTICATION gkrkddPd112!
```

```
docker run -d --name redis-container -p 6379:6379 -v redis
```

## Docker 및 Jenkins 설정

frontend와 backend 각 폴더 루트에 Dockerfile, Jenkinsfile 생성

### backend

#### Dockerfile

```
FROM openjdk:17
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Dspring.profiles.active=prod","-jar",
// "-Dspring.profiles.active=prod" : 로그백 설정을 위한 프로필
```

#### Jenkinsfile

```
pipeline {
    agent any
    environment {
        AWS_PUBLIC_IP = '172.26.8.148'
        SSH_CMD = 'ssh -i /var/jenkins_home/.ssh/id_rsa ubuntu@AWS_PUBLIC_IP'
        DOCKER = 'sudo docker'
```



```

        repository = "sejinnnnnn/pawsitive_backend" //docke
        DOCKERHUB_CREDENTIALS = credentials('dockerhub') //
        dockerImage = ''
        REPO = "s10-webmobile1-sub2/S10P12C111"
    }

    stages {

        stage('Build') {
            steps {
                dir('./backend') {
                    script {
                        sh "chmod +x ./gradlew"
                        sh './gradlew clean build'
                    }
                }
            }
        }

        stage('SonarQube analysis') {
            steps {
                withSonarQubeEnv('SonarQube-Server') {
                    dir('./backend') {
                        sh './gradlew sonarqube'
                    }
                }
            }
        }

        stage('Build image') {
            steps {
                script {
                    dir('./backend') {
                        dockerImage = docker.build repository
                    }
                }
            }
        }
    }

```

```

    }
}

stage('Login'){
    steps{
        sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login'
    }
}

stage('Push image') {
    steps {
        script {
            sh 'docker push $repository' //docker push
        }
    }
}

stage('Clean image') {
    steps {
        sh 'docker rmi $repository' // jenkins 컨테이너에 있
    }
}

stage('Deployment') {
    steps {
        script {
            try {
                sh '$SSH_CMD $DOCKER stop pawsitive_backend'
                sh '$SSH_CMD $DOCKER rm pawsitive_backend'
            } catch (e) {
                sh 'echo "fail to stop and remove container"'
            }
        }
        sh '$SSH_CMD $DOCKER pull $repository:latest'
        sh '$SSH_CMD $DOCKER run --name pawsitive_backend'
    }
}

```

```

    }
  }
}

```

## frontend

### Dockerfile

```

# 가져올 이미지를 정의
FROM node:20.11.0
# 경로 설정하기
WORKDIR /frontend
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.
COPY . .
# 3000번 포트 노출
EXPOSE 3000
# npm start 스크립트 실행
CMD ["npm", "run", "dev"]

```

### Jenkinsfile

```

pipeline {
  agent any
  environment {
    AWS_PUBLIC_IP = '172.26.8.148'
    SSH_CMD = 'ssh -i /var/jenkins_home/.ssh/id_rsa ubuntu@'
    DOCKER = 'sudo docker'
    repository = "sejinnnnnn/pawsitive_frontend" //dockerhub에 있는 이미지
    DOCKERHUB_CREDENTIALS = credentials('dockerhub') //dockerhub에 있는 자격증명
    dockerImage = ''
    REPO = "s10-webmobile1-sub2/S10P12C111"
  }
}

```

```

stages {

    stage('Install') {
        steps {
            dir('./frontend') {
                script {
                    sh 'npm install'
                }
            }
        }
    }

    stage('Build image') {
        steps {
            script {
                dir('./frontend') {
                    dockerImage = docker.build repository
                }
            }
        }
    }

    stage('Login'){
        steps{
            sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login'
        }
    }

    stage('Push image') {
        steps {
            script {
                sh 'docker push $repository' //docker push
            }
        }
    }
}

```

```

stage('Clean image') {
    steps {
        sh 'docker rmi $repository' // jenkins 컨테이너에 있
    }
}

stage('Deployment') {
    steps {
        script {
            try {
                sh '$SSH_CMD $DOCKER stop pawsitive_frontend'
                sh '$SSH_CMD $DOCKER rm pawsitive_frontend'
            } catch (e) {
                sh 'echo "fail to stop and remove container"'
            }
        }
        sh '$SSH_CMD $DOCKER pull $repository:latest'
        sh '$SSH_CMD $DOCKER run --name pawsitive_frontend'
    }
}
}
}

```

## 로컬 실행 가이드

1. 프로젝트를 클론 합니다.

```
$ git clone https://lab.ssafy.com/s10-webmobile1-sub2/S10I
```

2. 프로젝트 실행을 합니다.

- front

```
$ cd frontend
$ npm i
$ npm run dev
```

- back

```
$ cd backend/build/libs
$ java -jar backend-0.0.1-SNAPSHOT.jar
```

3. front 서버는 <http://localhost:3000/> api 서버는 <http://localhost:8080/> 에서 확인합니다.

- swagger :

<http://localhost:8080/swagger-ui/index.html>

---

## 외부서비스

### OpenVidu 배포 및 설정

1. 해당 포트들을 방화벽 해제를 한다.

- 22 TCP: to connect using SSH to admin OpenVidu.
- 80 TCP: if you select Let's Encrypt to generate an SSL certificate this port is used by the generation process.
- 443 TCP: OpenVidu server and application are published by default in standard https port.
- 3478 TCP+UDP: used by STUN/TURN server to resolve clients IPs.
- 40000 - 57000 TCP+UDP: used by Kurento Media Server to establish media connections.
- 57001 - 65535 TCP+UDP: used by TURN server to establish relayed media connections.

- 80, 443, 3478, 5442, 5443, 6379, 8888 포트를 사용할 수 있는 상태여야 한다.

→ 이미 nginx를 올린 상태라면 잠시 중단하고 openvidu 서버부터 올릴 것

2. openvidu 설치를 한다.

```
$ sudo su
$ cd /opt
```

```
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io,
```

### 3. .env 작성 후 서버 시작

```
$ cd openvidu
$ nano .env
$ ./openvidu start
```

.env

```
DOMAIN_OR_PUBLIC_IP=i10c111.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server
OPENVIDU_SECRET=PAWSITIVE

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

# If CERTIFICATE_TYPE=letsencrypt, you need to configure :
LETSENCRYPT_EMAIL=pawsitiver24@gmail.com
```

4. <https://i10c111.p.ssafy.io> 에 접속하여 정상적으로 작동하는지 확인한다.
5. 정상적으로 작동한다면 아래의 명령어를 통해 서버를 중단시키고 포트 변경을 위해 .env를 변경한다.

```
$ ./openvidu stop
```

```
# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be  
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.  
# WARNING: the default port 80 cannot be changed during the deployment.  
# if you have chosen to deploy with the option CERTIFICATE_DOMAIN_NAME  
HTTP_PORT=8442
```

```
# Changes the port of all services exposed by OpenVidu.  
# SDKs, REST clients and browsers will have to connect to  
HTTPS_PORT=8443
```

6. 변경 후 아래의 명령어를 통해 시작 및 <https://i10c111.p.ssafy.io:8443/>에 접속해 정상적으로 동작하는지 확인한다.

```
$ ./openvidu start
```

이 화면이 나온다면 성공한 것!



## 카카오 지도

### Kakao Develop REST API (카카오 지도)

지도 서비스 및 주소 변환 API 사용

<https://developers.kakao.com/product/map>



1. Kakao developer에서 로그인 후 애플리케이션 추가
2. 내 애플리케이션에 프로젝트 pawsitive 추가



### 3. JavaScript API 사용



### 4. 사이트 도메인 등록

#### Web

사이트 도메인	https://i10c111.p.ssafy.io http://localhost:3000
---------	---

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

## 5. Frontend `.env` 파일 내 Kakao Map API를 위한 설정 추가

```
interface ImportMetaEnv {  
  readonly VITE_APP_KAKAO_MAP_API_KEY: string  
}  
  
interface ImportMeta {  
  readonly env: ImportMetaEnv  
}
```

## Teachable Machine

머신러닝 모델을 쉽고 빠르게 만들 수 있도록 제작된 웹 기반 도구

<https://teachablemachine.withgoogle.com/>

1. <https://teachablemachine.withgoogle.com/> 로 이동하여 이미지 프로젝트 선택
2. 클래스 생성하여 데이터로 학습
3. 모델 학습시키기 버튼 클릭
4. 학습 완료 후 업로드 (공유 가능한 링크) 선택 → 모델 업로드
5. 모델 업로드 후 공유 가능한 링크와 사용할 코드 스니펫을 제공해 줌. 이를 사용하여 리액트에서 개발

프로젝트에서 모델을 사용하려면 모델을 내보내세요.



Tensorflow.js ⓘ

Tensorflow ⓘ

Tensorflow Lite ⓘ

모델 내보내기:

☒ 업로드(공유 가능한 링크)

☐ 다운로드

모델 업로드

공유 가능한 링크:

[https://teachablemachine.withgoogle.com/models/\[...\]](https://teachablemachine.withgoogle.com/models/[...])

모델을 업로드하면 Teachable Machine에서 이 링크에 모델을 호스팅합니다. (FAQ: 내 모델을 사용할 수 있는 사용자는 누구인가요?)

모델에서 사용할 코드 스니펫:

Javascript

p5.js

Github에 참여

Learn more about how to use the code snippet on [github](#).

```
<div>Teachable Machine Image Model</div>
<button type="button" onclick="init()">Start</button>
<div id="webcam-container"></div>
<div id="label-container"></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-
image.min.js"></script>
<script type="text/javascript">
  // More API functions here:
  // https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image
```

복사