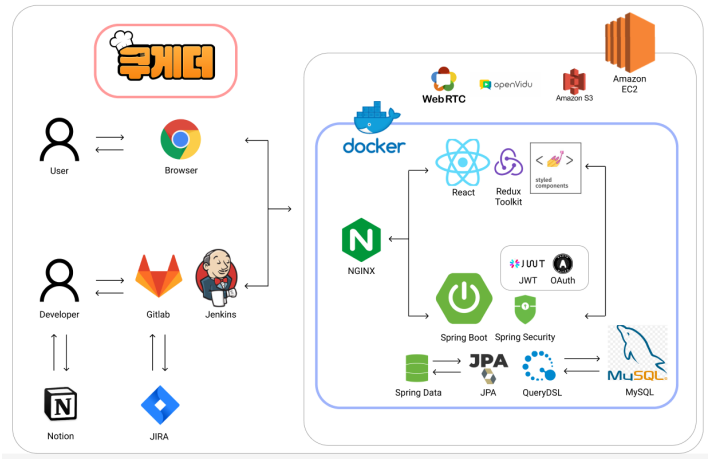
포팅 매뉴얼

SSAFY 8th 공통프로젝트

B206 쿠게더 : 포팅 매뉴얼

1. 프로젝트 기술 스택




```
FE : React(v17.0.2), redux: ^4.1.2, react-redux: ^8.0.5, React-router-dom(v5.3.4), styled-components(v5.3.6)
@reduxjs/toolkit: ^1.9.1

BE : Java(JDK 11), SpringBoot (2.7.7), SpringDataJPA, QueryDSL (5.0.0), Swagger (2.9.2)

DB : MySQL(v8.0.30)

Storage : S3 Bucket

WebRTC : openvidu(2.25.0)

IDE: IntelliJ, VSCode

DevOps : Docker(20.10.23), Jenkins(2.375.3), nginx(1.18.0), AWS ec2
```

2. 프론트엔드 배포

1) clone

```
git clone https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B206.git
```

2) build

```
docker build -t dev-front .
```

3) run

```
docker run --name react-container -d -p 3000:3000 dev-front
```

- Dockerfile

```
# 가져올 이미지를 정의
FROM node:14
# 경로 설정하기
WORKDIR /app
# 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 . .

# 각각의 명령어들은 한줄 한줄씩 캐싱되어 실행된다.
# package.json의 내용은 자주 바뀌진 않을 거지만
# 소스 코드는 자주 바뀌는데
# npm install과 COPY . . 를 동시에 수행하면
# 소스 코드가 조금 달라질때도 항상 npm install을 수행해서 리소스가 낭비된다.

# 3000번 포트 노출
EXPOSE 3000

# npm start 스크립트 실행
CMD ["npm", "start"]
```

3. 백엔드 배포

1) clone

```
git clone https://lab.ssafy.com/s08-webmobile1-sub2/S08P12B206.git
```

2) 권한 주기

```
chmod +x ./gradlew
```

3) build

```
./gradlew clean build
docker build -t dev-back .
```

4) run

```
docker run --name spring-container -d -p 9000:9000 dev-back
```

- Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

- docker-compose.yml

```
version: "3"
services:
  database:
    image: mysql
    environment:
      MYSQL_DATABASE: coogether
      MYSQL_USER: i8b206
      MYSQL_PASSWORD: ******
      MYSQL_ROOT_HOST: "%"
      MYSQL_ROOT_PASSWORD: ****
      TZ: Asia/Seoul
    volumes:
      - ./db/data:/var/lib/mysql
    ports:
      - 3306:3306
    restart: always
```

- docker-compose 실행 / 중지

```
//실행
docker-compose up --build -d

//중지
docker-compose down
```

4. Nginx 설치

1) Nginx 설치

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

2) 설치 확인

```
sudo nginx -v
```

3) Nginx 중지

```
sudo systemctl stop nginx
```

- nginx.conf

```
## 첫 줄 주석처리
# include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*.conf;
```

5. SSL 인증서 발급

1) Let’s Encrypt 설치

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

2) 인증서 적용 및 .pem 키

```
sudo letsencrypt certonly --standalone -d i8b206.p.ssafy.io
```

3) 발급 경로 확인

```
cd /etc/letsencrypt/live/i8b206.p.ssafy.io
```

4) .conf 파일 생성

```
cd /etc/nginx/sites-available

sudo vim test.conf
```

5) test.conf 작성

```
server {
        # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
        location /{
                proxy_pass http://localhost:3000;
        }

        # 백엔드 연결(포트 번호는 본인의 백엔드 포트번호를 입력)
        location /api {
                proxy_pass http://localhost:9000/api;
         }

    listen 443 ssl; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate /etc/letsencrypt/live/i8b206.p.ssafy.io/fullchain.pem; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate_key /etc/letsencrypt/live/i8b206.p.ssafy.io/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    # 도메인 이름을 입력
    if ($host = i8b206.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name i8b206.p.ssafy.io;
    return 404; # managed by Certbot
}
```

6) 심볼릭 링크 연결

```
sudo ln -s /etc/nginx/sites-available/test.conf /etc/nginx/sites-enabled/test.conf
```

7) 테스트

```
sudo nginx -t
```

8) Nginx 재시작

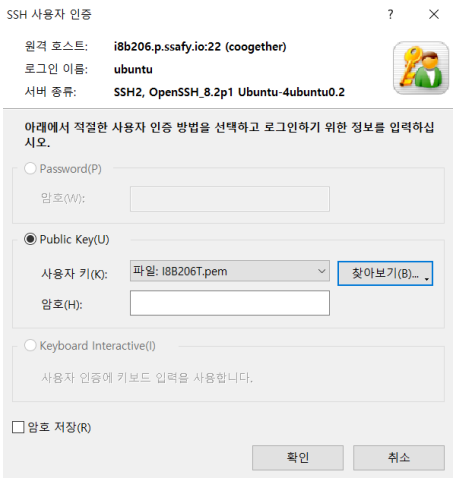
```
sudo systemctl restart nginx
```

9) Nginx 상태 확인

```
sudo systemctl status nginx
```

6. EC2 세팅

1) .pem키로 shell 접속



2) 관리자 모드로 전환

```
sudo su -
```

7. AWS S3 Bucket

https://s3.console.aws.amazon.com/s3/buckets

application.yml

```
cloud:
  aws:
    credentials:
      access-key:
      secret-key:
    s3: #버킷이름
      bucket:
    region: #S3 지역
      static: ap-northeast-2
    stack:
      auto: false
```

8. OPENVIDU

1) 설정

```
//.env
DOMAIN_OR_PUBLIC_IP=i8b206.p.ssafy.io
OPENVIDU_SECRET=
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL= gusrnas@gmail.com

HTTP_PORT=
HTTPS_PORT=
```

2) 실행

```
./openvidu start
```

9. Jenkins & Docker

1) 젠킨스 설치

```
# 젠킨스 설치
docker run --name jenkins-container -d -p 9999:8080 -p 50000:50000 -v /home/ubuntu/jenkins:/var/jenkins_home -v /var/run/docker.sock:/
```

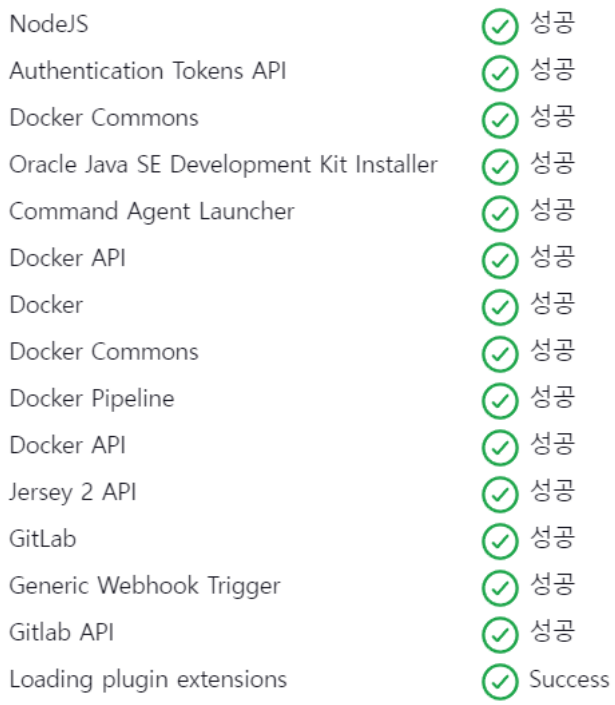
2) shell에서 젠킨스 접속

```
docker exec -it jenkins-container bash
```

3) docker-compose 설치

```
#docker compose 설치
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

docker ps

curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c

chmod +x /usr/local/bin/docker-compose

docker-compose -version
```

4) jenkins 플러그인 설치



5) Credentials 추가

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gusrnqq	gusrnqq@naver.com/******

KAKAO 소셜 로그인

Kakao Developers

카카오 API를 활용하여 다양한 어플리케이션을 개발해보세요. 카카오 로그인, 메시지 보내기, 친구 API, 인공지능 API 등을 제공합니다.

https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api

kakao developers

DB dump

Windows10 (C:) › SSAFY › S08P12B206 › exec › dump

dump 검색

이름	수정한 날짜	유형	크기
coogether_cooking_room	2023-02-16 오후 1:45	SQL Text File	8KB
coogether_follow	2023-02-16 오후 1:45	SQL Text File	3KB
coogether_history	2023-02-16 오후 1:45	SQL Text File	4KB
coogether_ingredient	2023-02-16 오후 1:45	SQL Text File	38KB
coogether_ingredient_fav	2023-02-16 오후 1:51	SQL Text File	2KB
coogether_ingredient_list	2023-02-16 오후 1:45	SQL Text File	76KB
coogether_my_ingredient_manage	2023-02-16 오후 1:50	SQL Text File	2KB
coogether_recipe	2023-02-16 오후 1:50	SQL Text File	263KB
coogether_recipe_step	2023-02-16 오후 1:49	SQL Text File	230KB
coogether_refresh_token	2023-02-16 오후 1:48	SQL Text File	2KB
coogether_report	2023-02-16 오후 1:47	SQL Text File	2KB
coogether_user	2023-02-16 오후 1:47	SQL Text File	3KB
coogether_user_join_list	2023-02-16 오후 1:47	SQL Text File	2KB