

VilEZ(빌리지) 배포 매뉴얼

본 문서는 안드로이드와 웹으로 개발된 VilEZ를 사용하기 위한 가이드를 안드로이드와 프론트 엔드 서버, QR 서버, 백엔드 스프링 서버를 배포하는 과정에 대해 서술하고 있습니다.

1. 안드로이드 배포

1.1. AOS 개발 환경

안드로이드

- Android Studio Dolphin (2021.3.1 Patch 1)
- targetSDK 32
- minSDK 26
- Kotlin

1.2. Android 기술 스택

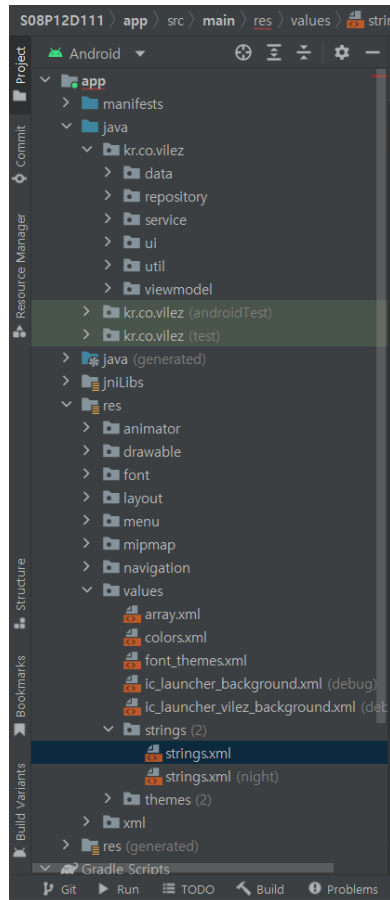
안드로이드

- Retrofit 2.9.0 (통신 라이브러리)
- OkHttp 4.10.0
- Dagger-Hilt (의존성 주입 라이브러리)
- JetPack Paging3 (페이징 라이브러리)
- Coroutines Flow (비동기 데이터 처리 라이브러리)
- Glide 4.14.2 (이미지 로드 라이브러리)
- ViewModel-ktx 2.5.1
- Live data 2.5.1
- Fragment-ktx 1.5.5
- Naver OAuth
- Kakao OAuth
- Navigation 2.3.5 (화면 전환, 스택 관리 라이브러리)
- Kakao map (지도 라이브러리)
- Material Calendar View
- Card View
-
- Stmop 2.0.5 (웹소켓 실시간 통신 라이브러리)
- OAuth (로그인 보안 라이브러리)
- MPAndroidChart 3.1.0 (통계 라이브러리)
- FCM (파이어베이스 푸시 알림 라이브러리)
- CalendarView (캘린더 라이브러리)
- Room (내부 데이터 베이스 라이브러리)
- Wearable Service 17.1.0 (폰, 워치 데이터 교환 라이브러리)

1.3. Android 포팅 가이드

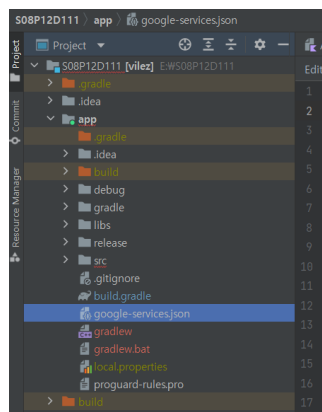
https://drive.google.com/file/d/1b6L_UzxUQC9TSaValOTI0EhWqmYe9O7I/view?usp=sharing

1.3.1. API 키 (naver, kakao)



- 네이버, 카카오 계정으로 로그인 하기 위해 사용
 - 단 카카오 로그인은 해시 키 보안으로 인해 안드로이드 스튜디오에서 앱을 설치할경우 사용 불가
 - 배포키를 발급받았기 때문에 apk로 배포시 정상 작동
- 카카오 지도 사용을 위함
- [app]-[src]-[main]-[res]-[values]-strings.xml 추가

1.3.2. google-service.json



- FCM 사용을 위한 파일
- 보기 모드를 Project로 변경
- [app]-[src]-google-services.json 추가

2. 백엔드 배포 과정

2.1. 개발환경 및 배포 환경

개발 환경

- Spring boot version : 2.7.7
- 빌드 도구 : spring-boot-maven-plugin
- IDE : IntelliJ IDEA 2022.3.1

배포 환경

- 배포 서버 : Ubuntu 20.04.4 LTS (AWS)
- Docker : 20.10.23
- Maven : Apache Maven 3.8.7

2.2. API 서버활성을 위한 배포과정

- 기본적으로 CI/CD툴의 젠킨스를 이용한다.
- 위 내용은 6번(젠킨스를 통한 자동배포) 항목에서 자세히 설명한다.

3. QR 페이지 배포

1. Git lab 의 <https://lab.ssfy.com/s07-webmobile4-sub2/S07P12D101> 위치로 이동하여 Clone한다. <FEDeploy> branch로 checkout 한다.
 - a. 특이사항 : FDDeploy 브랜치와 Deploy 브랜치는 동일한 브랜치로 간주할 수 있습니다. 하지만 저희조에서는 Jenkins를 이용한 재배포를 복습하기 위하여 따로 브랜치를 생성하였습니다. Deploy 브랜치로 해도 동일하겠으나 FEDeploy를 사용하는 것을 권장합니다.
2. 2번 API 서버활성을 위한 배포과정과는 다르게 react 작업 공간 root에 있는 Dockerfile을 사용하면 됩니다.

FROM node:16.14.0

WORKDIR /usr/src/app

COPY package.json ./

RUN npm install

RUN yarn global add serve

COPY ./ ./

1. 아래의 명령을 이용하여 컨테이너화 시킵니다. => docker build --tag feimg:fetag .
2. <도메인>/manager 로 접속, 소셜로그인
3. 신고글을 처리할 수 있는 페이지가 주어진다. .env 파일또한 포함되어있으므로 따로 설정할 필요가 없다.
4. 위 과정을 다 거치고 나면 서버의 구조는 밑에 그림과 같다.

4. 프론트엔드 배포

4.1. 개발 환경

- Node : v8.19.2

- React : v18.2.0
- Javascript

4.2. 기술 스택

Package.json 에는 다음과 같은 내용들이 있어야 한다.

```
{
  "name": "vilez",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/eslint-plugin": "^11.10.0",
    "@emotion/react": "^11.10.5",
    "@stomp/stompjs": "^6.1.2",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.2.2",
    "qrcode.react": "^3.1.0",
    "react": "^18.2.0",
    "react-animated-css": "^1.2.1",
    "react-datepicker": "^4.8.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.7.1",
    "react-router-dom": "^6.6.2",
    "react-scripts": "5.0.1",
    "react-signature-canvas": "^1.0.6",
    "recoil": "^0.7.6",
    "recoil-persist": "^4.2.0",
    "sockjs-client": "^1.6.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@babel/eslint-parser": "^7.19.1",
    "@babel/preset-react": "^7.18.6",
    "eslint": "^8.31.0",
    "eslint-config-prettier": "^8.6.0",
    "eslint-plugin-prettier": "^4.2.1",
    "eslint-plugin-react": "^7.32.0",
    "prettier": "^2.8.3"
  }
}
```

- name : 필수로 입력돼야 하는 사항이며 서비스의 이름이된다.
- version : 필수로 입력돼야 하는 사항이며 서비스의 버전이다.
- private : 패키지를 비공개 할 건지 여부를 표시한다.
- dependencies : 의존하고 있는 라이브러리와 각 라이브러리들의 설치된 버전을 표시한다.
 - @emotion/eslint-plugin : emotion 에 대한 eslint 규칙을 설정하는 라이브러리이다.
 - @emotion/react : react에서 사용하는 css 스타일링 라이브러리이다.

- @stomp/stompjs : 서비스의 실시간 채팅과 실시간 공유지도 기능을 사용하기 위해 설치한 stomp 브로커이다.
- @testing-library/jest-dom : 테스트 코드를 작성하기 위한 CRA로 react를 생성했을 시, 자동으로 생성되는 testing library 패키지이다.
- @testing-library/react : 테스트 코드를 작성하기 위한 CRA로 react를 생성했을 시, 자동으로 생성되는 testing library 패키지이다.
- @testing-library/user-event : 테스트 코드를 작성하기 위한 CRA로 react를 생성했을 시, 자동으로 생성되는 testing library 패키지이다.
- axios : node.js와 브라우저를 위한 Promise기반의 HTTP 클라이언트이다.
- qrcode.react : DOM에 렌더링하기 위한 QR코드를 생성하는 라이브러리이다.
- react : 사용자 인터페이스를 만들기 위한 JavaScript 라이브러리
- react-animated-css : animated.css를 사용하여 리액트에서 요소에 애니메이션을 주는 라이브러리이다.
- react-datepicker : 서비스의 캘린더에 여러 기능을 부여하기 위해 사용한 라이브러리이다.
- react-dom : react에서 작성한 여러 컴포넌트를 html과 연결하는 작업을 해주는 라이브러리이다.
- react-icons : 서비스에 다양한 아이콘을 적용하기 위해 사용한 라이브러리이다.
- react-router-dom : react에서 라우팅을 할 수 있게 해주는 라이브러리이다.
- react-scripts : CRA로 react 프로젝트를 만들었을 때, 사용하는 스크립트 및 구성이 포함되어 있다.
- react-signature-canvas : react에서 canvas를 사용하여 서명을 할 수 있게 해주는 라이브러리이다.
- recoil : 서비스에서 전역으로 상태를 관리하기 위해 사용한 라이브러리이다.
- recoil-persist : recoil의 값을 localStorage에 저장시켜 영구 저장을 하기 위해 사용한 라이브러리이다.
- sockjs-client : 서비스의 실시간 채팅과 실시간 공유지도 기능을 구현하기 위해 사용했으며, websocket과 유사한 객체를 제공하는 javascript 라이브러리이다.
- web-vitals : 실제 사용자에게 대한 모든 성능 지표 측정 항목을 크롬에서 측정하는 도구이다.
- scripts : 여러 가지 npm 명령어를 설정하여 콘솔에서 사용할 수 있다.
 - start : 서비스를 시작할 때 사용하는 명령어이다.
 - build : 패키지를 빌드할 때 사용하는 명령어이다.
 - test : 패키지를 테스트할 때 사용하는 명령어이다.
 - eject : 빌드 구성이 만족스럽지 않을 때, 프로젝트에서 단일 빌드 종속성을 제거하는 명령어이다.
- eslintConfig : eslint 설정에 대해 정리한다.
 - extends : eslint가 어떤 설정을 따르는지 표시한다.
 - react-app : react 프로젝트에 eslint를 설정한다.
 - react-app/jest : react 프로젝트에 eslint를 설정한다.
- browserslist : 브라우저를 선택하는 옵션 기능만 따로 뽑아 놓은 도구이다.
 - production : 제품 빌드 시 지원하는 브라우저에 대한 설정이다.
 - >0.2% : 전 세계 점유율 0.2% 이상의 브라우저만 지원한다.
 - not dead : 지원이 중단되지 않은 브라우저만 지원하겠다.
 - not op_mini all : 오페라 미니는 제외한다.
 - development : 개발 시에는 아래와 같은 브라우저를 사용할 수 있다.
 - last 1 chrome version : 크롬의 최신버전을 사용한다.
 - last 1 firefox version : 파이어폭스의 최신버전을 사용한다.
 - last 1 safari version : 사파리의 최신버전을 사용한다.
- devDependencies : 개발 모드일때만 의존하는 라이브러리와 각 라이브러리들의 설치된 버전을 표시한다.
 - @babel/eslint-parser : eslint에서 babel parser 기능을 제공하는 라이브러리이다.

- @babel/preset-react : 리액트 애플리케이션을 만들 때 필요한 플러그인들의 집합으로, JSX로 작성된 코드들을 createElement 함수를 이용한 코드로 변환해 주는 바벨 플러그인이 내장되어 있다.
- eslint : javascript 문법에 오류가 없는지 검사하기 위해 사용하는 라이브러리이다.
- eslint-config-prettier : 불필요하거나 Prettier와 충돌할 수 있는 모든 규칙을 해제한다.
- eslint-plugin-prettier : eslint 규칙으로 prettier를 실행하는 라이브러리이다.
- eslint-plugin-react : eslint 규칙으로 react를 실행하는 라이브러리이다.
- prettier : 코드의 포매팅을 맞출 수 있는 라이브러리이다.

4.3 프론트엔드 서버활성을 위한 배포 과정

- 기본적으로 CI/CD툴의 젠킨스를 이용한다.
- 위 내용은 6번(젠킨스를 통한 자동배포) 항목에서 자세히 설명한다.

4.4 앱 키

```
REACT_APP_JS_APP_KEY=
REACT_APP_KAKAO_REST_API_KEY=
REACT_APP_NAVER_REST_API_KEY=
REACT_APP_API_BASE_URL=https://i8d111.p.ssafy.io/vilez
REACT_APP_KAKAO_REDIRECT_URI=
REACT_APP_NAVER_REDIRECT_URI=
```

.env 파일에 서비스에서 사용되는 중요한 키를 저장한다.

React 라이브러리를 사용하므로 키 이름의 앞에 REACT_APP을 붙여준다.

- `REACT_APP_JS_APP_KEY` : kakao map api를 사용할 때 쓰이는 key이다.
- `REACT_APP_KAKAO_REST_API_KEY` : kakao 소셜 로그인을 할 때 사용되는 key이다.
- `REACT_APP_NAVER_REST_API_KEY` : naver 소셜 로그인을 할 때 사용되는 key이다.
- `REACT_APP_API_BASE_URL` : 서비스 내에서 백엔드와 소통하는 REST API 요청의 엔드포인트이다.
- `REACT_APP_KAKAO_REDIRECT_URI` : kakao 소셜 로그인을 진행할 때, 인가 코드를 받는 uri이다.
- `REACT_APP_NAVER_REDIRECT_URI` : naver 소셜 로그인을 진행할 때, 인가 코드를 받는 uri이다.

5. 외부 서비스에 대한 설명

5.1. Naver Oauth , Kakao Oauth

- `NaverOAuthServiceImpl` , `KakaoOAuthServiceImpl` 설정

```
String clientId = "";
String clientScret = "";
String state = "";
```

- `RestTemplate` 사용으로 카카오 서버와 통신
- 공식 문서
 - <https://developers.kakao.com/docs/latest/ko/kakaologin/common>
 - <https://developers.naver.com/docs/login/api/api.md>

5.2. FCM

```
graph TD
```

Mermaid --> Diagram

```
{
  "type": "service_account",
  "project_id": "vilez-4beed",
  "private_key_id": "2e334d4f94eeca6892c0c36c619b995b4b596b63",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQCmFgcYXR6xVLhR\n5LPQu/4gtPEf5syZ\n\"client_email\": \"firebase-adminsdk-6b9ky@vilez-4beed.iam.gserviceaccount.com\",
  \"client_id\": \"103360724558534392100\",
  \"auth_uri\": \"https://accounts.google.com/o/oauth2/auth\",
  \"token_uri\": \"https://oauth2.googleapis.com/token\",
  \"auth_provider_x509_cert_url\": \"https://www.googleapis.com/oauth2/v1/certs\",
  \"client_x509_cert_url\": \"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-6b9ky%40vilez-4beed.iam.gserviceac\n\"}
}
```

- firebase_service_key.json에는 상기와 같은 내용이 포함됨

- <https://console.firebase.google.com/> 구글 파이어베이스 콘솔에서 vilez 프로젝트에 접속
- [프로젝트 설정] 메뉴의 [서비스 계정]탭에서 자바 언어로 비공개 키를 생성
- 생성된 파일을 Spring Boot의 [프로젝트]-[src]-[main]-[resources]-[firebase]-firebase_service_key.json 경로에 추가

5.3. Naver Object Storage

[SpringBoot] SpringBoot를 이용한 AWS S3에 여러 파일 업로드 및 삭제 구현하기 — ROOPRETELCHAM (tistory.com)

<https://velog.io/@whitebear/Spring에서-Naver-Cloud-Object-Storage-사용할-때-에러-사냥하기>

6. 젠킨스를 통한 자동배포

6.1. 초기 세팅

도커를 통하여 젠킨스를 설치한다.

- <https://dongle94.github.io/docker/docker-ubuntu-install/>
- 해당 사이트를 참고하여 도커 설치한다.
- 설치 후 다음 명령어를 통하여 젠킨스를 설치한다.

Docker Out Of Docker(DooD) 기법으로 도커를 설치한다.

```
docker run --privileged --name docker-server -itd -p 10022:22 -p 8081:8080 -e container=docker -v /sys/fs/cgroup:/sys/fs/cgroup edowon
```

도커에 Ansible을 설치한다.

- 명령어를 통하여 Ansible을 설치한다.

```
docker run -itd --name ansible-server -p 20022:22 -e container=docker --tmpfs /run --tmpfs /tmp -v /sys/fs/cgroup:/sys/fs/cgroup:ro -v
```

젠킨스 플러그인을 설치한다.


- Ansible
- GitLab
- Publish Over SSH

젠킨스에 Ansible 컨테이너를 연결한다.

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

 Concealed

Path to key ?

Key ?

☐ Disable exec ?

SSH Servers

SSH Server Name ?

ansible-server

Hostname ?

172.26.5.131

Username ?

root

Remote Directory ?

.

▼ 젠킨스 설정

- JDK

JDK

JDK installations

List of JDK installations on this system

Add JDK

JDK

Name

jdk11.0.18

JAVA_HOME

/opt/java/openjdk

☐ Install automatically ?

- Maven

Maven

Maven installations

List of Maven installations on this system

Add Maven

Maven

Name

maven3.8.7

☒ Install automatically ?

Install from Apache

Version

3.8.7

Add Installer ▾

- Node

NodeJS

Name

Node

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 19.5.0

For the underlying architecture, if available, force the installation of the 32bit package. Other

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the pac

npm install -g serve

Global npm packages refresh hours

Duration, in hours: halve 2 npm cache update. Note that 0 will always update npm cache

GitLab Webhook 등록

- 젠킨스 프로젝트 설정에서 Build Triggers에 webhook을 등록한다.
- Push Events을 체크한다.

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://i8d111.p.ssafy.io:8080/project/Back-End-Project> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☐ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

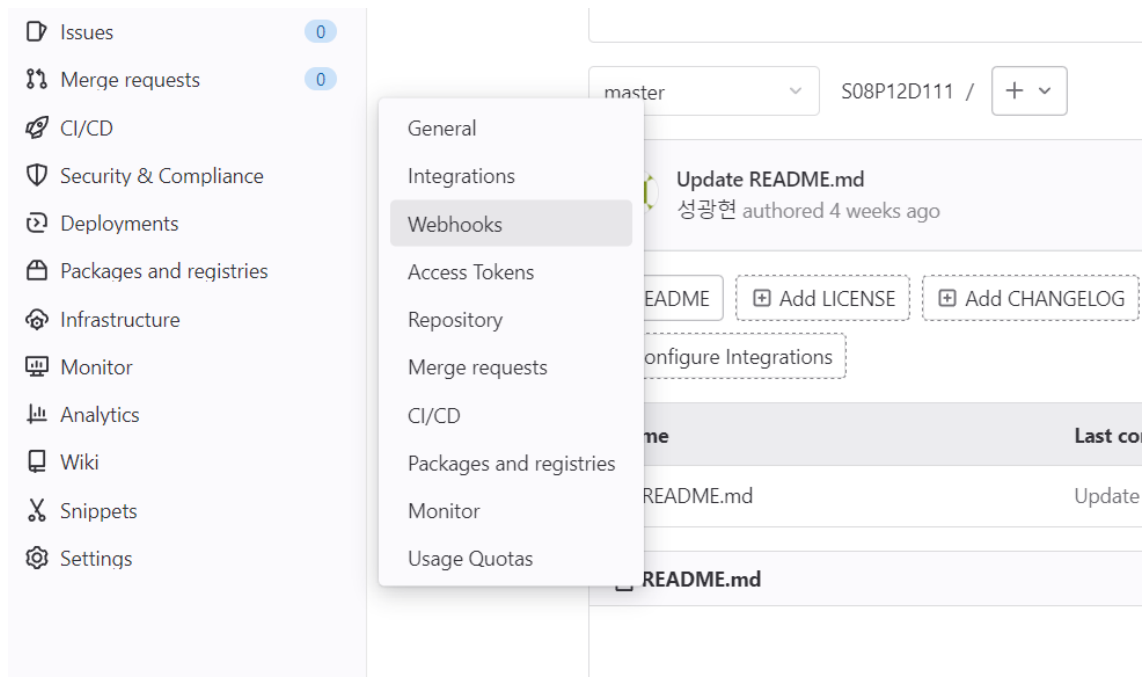
Rebuild open Merge Requests

- Secret Key를 발급한다.

Secret token ?

124c03 [REDACTED] 6bb2

- <https://lab.ssafy.com/s08-webmobile2-sub2/S08P12D111>에서 Settings의 Webhooks를 누른다.



- Webhooks 에 Jenkins URL을 적고 Push events에 fe/develop, mo/develop, be/develop, qr/develop중 등록할 브랜치 이름을 적는다.
그리고 Jenkins에서 발급받은 Secret token을 적는다.

Q Search page

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the **X-Gitlab-Token** HTTP header.

Trigger

☒ Push events

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

- Add Webhooks 를 눌러 GitLab에 등록한다.

SSL verification

☒ Enable SSL verification

[Add webhook](#)

6.2. Ansible 설정

▼ 파일 목록

- Infra의 Ansible 서버

```
[root@43cee30467cc ~]# ls
anaconda-ks.cfg      check.tar             create-cicd-project-container-playbook.yml  Dockerfile2      vilez.tar
anaconda-post.log    cp                   create-cicd-project-image-playbook.yml      Dockerfile3      vilez.war
catalina1.sh         create-cicd-front-container-playbook.yml     create-cicd-project-qr-playbook.yml         original-ks.cfg
catalina.sh          create-cicd-front-image-playbook.yml          Dockerfile
```

- 사용 네트워크 및 컨테이너

```
{
  "containers": [
    {
      "id": "01463813303c7f3e3c7434077476a406fa11cc348f23d268d1c682c5c8a0914c", {
        "Name": "jenkins-server",
        "EndpointID": "df2c1bf41ee0b51cd5d37e8da7f1e89a2b76aade9d45d717943d85cd0fdd",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    },
    {
      "id": "8d7417fc349b6973f9fec363728735a739497ac50fead774645c46554c31", {
        "Name": "docker-server",
        "EndpointID": "2f5b900b2a7c949b641b3a7c2b71930a66820eFdf53145464c1240d434ad7440",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
      }
    },
    {
      "id": "30ee30487ccff1c61d3d0188651e85a06dc1e8a87886b8dec405e031b2b65791", {
        "Name": "ansible-server",
        "EndpointID": "2f5b900b2a7c949b641b3a7c2b71930a66820eFdf53145464c1240d434ad7440",
        "MacAddress": "02:42:ac:11:00:04",
        "IPv4Address": "172.17.0.4/16",
        "IPv6Address": ""
      }
    },
    {
      "id": "141e85f49245a4e39046313f57e3970b0fdd42fe0b322f88468e3db2eff9", {
        "Name": "vilezfe",
        "EndpointID": "bda3d247e8ff8e49c7d21c00eefc250af4717f292a807604f62e3c1b008de39",
        "MacAddress": "02:42:ac:11:00:06",
        "IPv4Address": "172.17.0.5/16",
        "IPv6Address": ""
      }
    },
    {
      "id": "7cfa7983fab4b7260b8267190e1c1029bec75e8e3e3a372c496484f517c500e", {
        "Name": "qr-server",
        "EndpointID": "2f5b900b2a7c949b641b3a7c2b71930a66820eFdf53145464c1240d434ad7440",
        "MacAddress": "02:42:ac:11:00:08",
        "IPv4Address": "172.17.0.6/16",
        "IPv6Address": ""
      }
    }
  ]
}
```

- Dockerfile, Dockerfile2, Dockerfile3
 - 각 사용이미지에 상세 기록 되어있음.
- create-cicd-front-image-playbook.yml

```
- hosts: all
#   become: true

tasks:
- name: remove the docker image from the ansible server
  command: docker rmi vilezfe
  ignore_errors: yes

- name: stop current running container
  command: docker stop vilezfe
  ignore_errors: yes

- name: remove stopped cotainer
  command: docker rm vilezfe
  ignore_errors: yes

- name: create a docker image with deployed waf file
  command: docker build -t vilezfe -f Dockerfile2 .
  args:
    chdir: /root

- name: create a container using cicd-project-ansible image
  command: docker run -d --name vilezfe -p 3000:3000 vilezfe
```

- 1행 hosts에 명시된 모든 도커 컨테이너에 적용
- 4행 작업 실행
 - 1번 작업 vilezfe의 이미지를 제거 (에러 무시)
 - 2번 작업 현재 vilezfe로 실행된 컨테이너 중지 (에러 무시)
 - 3번 작업 vilezfe의 이름을 가진 컨테이너 삭제
 - 4번 작업 Dockerfile2의 설정에 따라 vilezfe라는 이름으로 이미지 생성
 - 5번 작업 vilezfe의 이미지를 백그라운드로 3000포트를 사용하여 컨테이너 실행
- create-cicd-project-image-playbook.yml

```
- hosts: all
#   become: true

tasks:
- name: create a docker image with deployed waf file
  command: docker build -t gch03944/vilez .
  args:
    chdir: /root

- name: push the image on Docker Hub
  command: docker push gch03944/vilez

- name: remove the docker image from the ansible server
  command: docker rmi gch03944/vilez
  ignore_errors: yes
```

- 1행 host에 명시된 모든 도커 컨테이너에 적용
- 4행 작업 실행
 - 1번 작업 Dockerfile의 설정에 따라 gch03944/vilez라는 이름으로 이미지 생성
 - 2번 작업 해당 이미지를 도커허브(프라이빗)에 배포
 - 3번 작업 서버에 남아있는 gch03944/vilez의 이미지를 삭제 (에러 무시)
- create-cicd-project-container-playbook.yml

```
- hosts: all
#   become: true

tasks:
- name: stop current running container
  command: docker stop vilez
  ignore_errors: yes

- name: remove stopped cotainer
  command: docker rm vilez
  ignore_errors: yes

- name: remove current docker image
  command: docker rmi gch03944/vilez
  ignore_errors: yes

- name: pull the newest docker image from Docker Hub
  command: docker pull gch03944/vilez:latest

- name: create a container using cicc-project-ansible image
  command: docker run -d --name vilez -p 8080:8080 gch03944/vilez
```

- 1행 host에 명시된 모든 도커 컨테이너에 적용
- 4행 작업 실행
 - 1번 작업 현재 vilez로 실행된 컨테이너 중지 (에러 무시)
 - 2번 작업 vilez의 이름을 가진 컨테이너 삭제
 - 3번 작업 gch03944/vilez의 이미지를 삭제 (에러 무시)
 - 4번 작업 도커허브에 배포된 gch03944/vilez를 최신버전으로 이미지 설치
 - 5번 작업 gch03944/vilez의 이미지를 vilez라는 이름으로 8080포트를 사용하여 백그라운드로 구동
- create-cicc-project-qr-playbook.yml

```
- hosts: all
#   become: true

tasks:
- name: stop current running container
  command: docker stop qr-server
  ignore_errors: yes

- name: remove stopped cotainer
  command: docker rm qr-server
  ignore_errors: yes

- name: remove current docker image
  command: docker rmi qr
  ignore_errors: yes

- name: build a docker image with deployed war file
  command: docker build -t qr -f Dockerfile3 .
  args:
    chdir: /root

- name: create a container using cicc-project-ansible image
  command: docker run -d --name qr-server -p 8089:8080 qr
```

- 1행 hosts에 명시된 모든 도커 컨테이너에 적용
- 4행 작업 실행
 - 1번 작업 qr-server로 실행된 컨테이너 중지 (에러 무시)
 - 2번 작업 qr-server의 이름을 가진 컨테이너 삭제 (에러 무시)
 - 3번 작업 qr-server의 이름을 가진 이미지 삭제 (에러 무시)
 - 4번 작업 Dockerfile3의 설정에 따라 qr 이라는 이름으로 이미지 생성
 - 5번 작업 qr의 이미지를 백그라운드로 8089포트를 사용하여 컨테이너 실행
- hosts

```
172.17.0.3
172.17.0.4
```

6.3. Nginx 설정

▼ 설명 및 내용

- openvidu기반 nginx
- custom-nginx.conf
- Reverse Proxy 설정 내용

```
# Your App
location /vilez {
    proxy_pass http://localhost:8081; # Openvidu call by default
}

location / {
    allow all;
    deny all;
    proxy_pass http://localhost:3000;
}

#####
# OpenVidu Locations  #
#####
#####
# Common rules  asbg  #
#####
# Dashboard rule
location /dashboard {
    allow all;
    deny all;
    proxy_pass http://openviduserver;
}

location /server/ {
    proxy_pass http://localhost:8089/server/index.html;
}

location /check/ {
    proxy_pass http://localhost:8089/check/index.html;
}
```

- /vilez
 - 백엔드 Spring 서버
- /
 - 프론트엔드 Node 서버
- /dashboard
 - 사용 X
- /server/
 - QR인증 페이지
- /check/
 - QR확인 페이지

6.4. 백엔드 서버 배포

- 젠킨스 파이프라인

```
pipeline {
    agent any
    tools {
        maven 'maven3.8.7'
    }
    stages {
        stage('github clone') {
            steps {
                git branch: 'be/develop', credentialsId: 'gch03944', url: 'https://lab.ssfy.com/s08-webmobile2-sub2/S08P12D111'
            }
        }

        stage('build'){
            steps {
                sh '''
                    echo build start
                    mvn clean compile package -DskipTests=true
                    '''
            }
        }

        stage('ssh') {
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false,
                    ansible-playbook -i ../etc/ansible/hosts create-cicd-project-container-playbook.yml --limit 172.17.0.3'', execTimeout: 120000, fl
```

```
}
}
```

- 서비스 백엔드 서버
- Maven 프로젝트(Spring Boot)
- Dockerfile 내용

```
FROM tomcat:9.0
LABEL org.opencontainers.image.authors="gch03944@gmail.com"
COPY ./vilez.war /usr/local/tomcat/webapps
```

- 1행 tomcat 9 버전을 구성하는 이미지를 만든다.
- 2행 저작권자 작성
- 3행 ./vilez.war 파일을 /user/local/tomcat/webapps에 복사한다.

6.5. 프론트 서버 배포

- 젠킨스 파이프라인

```
pipeline {
  agent any
  tools {
    nodejs 'Node'
  }
  stages {
    stage('github clone') {
      steps {
        git branch: 'fe/develop', credentialsId: 'gch03944', url: 'https://lab.ssfy.com/s08-webmobile2-sub2/S08P12D111'
      }
    }
    stage('build'){
      steps {
        sh '''
          echo build start
          npm install
          npm run build
          tar cvf vilez.tar build
          '''
      }
    }
    stage('ssh') {
      steps {
        sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false
      ]
    )
  }
}
```

- 서비스 프론트 서버
- Node기반 리엑트 라이브러리 서버
- Dockerfile 내용

```
FROM node:18
LABEL org.opencontainers.image.authors="gch03944@gmail.com"
ADD ./vilez.tar .
RUN ["node", "-v"]
RUN ["npm","install","-g", "serve"]
CMD ["serve","-s","./build"]
```

- 1행 nodejs 18버전을 구성하는 이미지를 만든다

- 2행 저작권자 작성
- 3행 디렉토리의 vilez.tar파일을 .디렉토리에 복사 및 압축 해제한다.
- 4행 이미지를 빌드할 때, node가 설치 되어있는지 -v를 통하여 버전을 확인한다.
- 5행 이미지를 빌드할 때, npm의 install를 사용하여 글로벌로 serve를 다운받는다.
- 6행 컨테이너를 최초 실행할 때, serve의 -s 명령어를 통하여 ./build를 서버로 실행한다.

6.6. QR 서버 배포

- 젠킨스 파이프라인

```
pipeline {
    agent any
    tools {
        maven 'maven3.8.7'
    }
    stages {
        stage('github clone') {
            steps {
                git branch: 'qr/develop', credentialsId: 'gch03944', url: 'https://lab.ssfy.com/s08-webmobile2-sub2/S08P12D111'
            }
        }
        stage('build'){
            steps {
                sh '''
                    tar -cvf server.tar server
                    tar -cvf check.tar check
                '''
            }
        }
        stage('ssh') {
            steps {
                sshPublisher(publishers: [sshPublisherDesc(configName: 'ansible-server', transfers: [sshTransfer(cleanRemote: false
```

- 서비스 QR서버
- Vanilla JS 기반 페이지
- Dockerfile 내용

```
FROM tomcat:9.0

LABEL org.opencontainers.image.authors="gch03944@gmail.com"

ADD ./server.tar /usr/local/tomcat/webapps
ADD ./check.tar /usr/local/tomcat/webapps
```

- 1행 tomcat 9 버전을 구성하는 이미지를 만든다.
- 2행 저작권자 작성
- 3행 ./server.tar 파일을 /user/local/tomcat/webapps에 복사한 후 압축 해제한다.
- 4행 ./check.tar 파일을 /user/local/tomcat/webapps에 복사한 후 압축 해제한다.

6.7. 주요 계정 및 프로퍼티가 정의된 파일 목록

- 백엔드 : application.properties
- 프론트엔드 : .env