readed 포팅 매뉴얼

날짜	@August 13, 2023

1. 개발환경

1.1. Frontend

- Node.js 18.17.0
- React 17.0.0

1.2. Backend

- Java
  - Java OpenJDK 1.8.0
  - Spring Boot 2.7.7
    - Spring Data JPA 2.7.6
    - Spring Security 5.7.6
    - JUnit 4.13.2
    - Lombok 1.18.24
    - Swagger 3.0.0
  - Gradle 7.6

1.3. Server

- Ubuntu 20.04 LTS
- Docker 24.0.5
- Docker Compose version v2.20.2
- Nginx 1.18.0-0ubuntu1.4
- OpenVidu

1.4. Database

- MySQL8.0.33
- Redis
- H2 1.4.200

1.5. UI/UX

- Figma

1.6. IDE

- Visual Studio Code
- IntelliJ IDEA
- Dbeaver

1.7. 형상 / 이슈관리

- Gitlab
- Jira

1.8. 기타 툴

- Posrman 버전?
- Figma
- Notion

2. EC2 세팅

EC2 접속

- I9A507T.pem 키가 있는 디렉토리에서 접속 명령어
- `ssh -i i9A507T.pem ubuntu@i9a507.p.ssafy.io`

Docker Engine 설치

- 참고 : 도커 공식문서 https://docs.docker.com/engine/install/ubuntu/

```
# 구 버전 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

# apt 패키지 업데이트
sudo apt-get update

# Docker Engine 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-com
pose-plugin
```

- `apt-get remove docker docker-engine docker.io containerd runc` : docker, docker-engine, docker.io, containerd, runc 중 설치된 게 있으면 삭제
- `apt-get update` : apt-get은 Advanced Packing Tool을 말한다. 관련된 라이브러리들을 update된 최신 정보로 가져온다

Docker-Compose 설치

- 참고 : 도커 공식 문서 https://docs.docker.com/compose/install/linux/

```
# apt 패키지 업데이트 후 최신버전 다운 (Ubuntu)
sudo apt-get update
sudo apt-get install docker-compose-plugin

# manually install
DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
mkdir -p $DOCKER_CONFIG/cli-plugins
curl -SL https://github.com/docker/compose/releases/download/v2.20.3/docker-compose-linux-
x86_64 -o $DOCKER_CONFIG/cli-plugins/docker-compose
```

```
# 버전확인
docker compose version
```

환경 설정

- 사용자 설정

```
# ssafy 계정 생성
sudo adduser ssafy

# docker 그룹에 ssafy 추가
sudo usermod -aG docker ssafy

# ssafy 계정으로 전환
su - ssafy
```

nginx.conf 파일 생성

- home/ssafy/readed/proxy에 nginx.conf 생성
- ssl 인증 전에는 인증서 파일 경로, 개인키 파일 경로 주석처리

```
# nginx가 리버시 프록시 역할을 하도록 nginx 파일 설정
user  nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    # upstream 설정은 docker-compose에서 설정한 서비스명 사용
    # docker-compose.yml에서 올라가는 컨테이너명으로 작성
    # 백엔드 upstream 설정
    upstream server  {
        # http:// 붙이면 안 됨
        server 3.38.252.22:8081;
        # 접속시 커넥션 유지 시간을 지정
        keepalive 1024;
    }

    # 프론트엔드 upstream 설정
    upstream client {
```

```
        server 3.38.252.22:3000;
    }

    server {
        # nginx를 통해 외부로 노출되는 port
        # http 80으로 진입해도 https 443로 리다이렉트
        listen 80;

        # 지정한 서버인증서에 포함된 도메인
        server_name i9a507.p.ssafy.io;
        server_tokens off;

        location / {
            return 301 https://$host$request_uri;
            # openvidu용 웹소켓 추가
            proxy_set_header Upgrade "websocket";
            proxy_set_header Connection "Upgrade";
            proxy_http_version 1.1;
        }

        # certbot 설정파일
        location /.well-known/acme-challenge/ {
            allow all;
            root /var/www/certbot;
        }

    }

    server {
        # default_server 필요 없음
        listen 433 ssl;

        # 인증서 파일 경로
        ssl_certificate /etc/letsencrypt/live/i9a507.p.ssafy.io/fullchain.pem;
        # 개인키 파일 경로
        ssl_certificate_key /etc/letsencrypt/live/i9a507.p.ssafy.io/privkey.pem;

        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # /api 경로로 오는 요청을 백엔드 upstream 의 /api 경로로 포워딩
        location /api/ {
            proxy_pass         http://3.38.252.22:8081;
        }
        # / 경로로 오는 요청을 프론트엔드 upstream 의 / 경로로 포워딩
        location / {
            proxy_pass         http://3.38.252.22:3000;
            # 시간 넉넉하게
            proxy_connect_timeout 300s;
            proxy_read_timeout 600s;
            proxy_send_timeout 600s;
            proxy_buffers 8 16k;
            proxy_buffer_size 32k;
            # openvidu용 웹소켓 추가
```

```
            proxy_set_header Upgrade "websocket";
            proxy_set_header Connection "Upgrade";
            proxy_http_version 1.1;
        }
    }

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;
    include /etc/nginx/conf.d/*.conf;
}
```

docker-compose 파일 작성

- 도커 이미지 빌드 및 컨테이너 실행 자동화 설정의 편리성을 위해 docker-compose 파일 생성
- home/ssafy/readed에 docker-compose.yml 생성
- jenkins, nginx, certbot, redis 설치

```
version: "3.0"
services:
  jenkins:
    image: jenkins/jenkins:lts
    user: root
    ports:
      - 8080:8080
    volumes:
      - /jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock

  nginx:
    image: nginx
    ports:
      - 80:80
      - 443:433
    volumes:
      - ./proxy/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    depends_on:
      - jenkins
    command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx
-g \"daemon off;\"'"
```

```
  certbot:
    image: certbot/certbot
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait
 $${!}; done;'"
    depends_on:
      - nginx

  redis:
    image: redis:latest
    ports:
      - 6379:6379
    volumes:
      - ./redis/data:/data
      - ./redis/conf/redis.conf:/usr/local/conf/redis.conf
    labels:
      - "name=redis"
      - "mode=standalone"
    restart: always
    command: redis-server /usr/local/conf/redis.conf
    depends_on:
      - jenkins
```

docker-compose 실행

- docker-compose.yml 파일이 위치한 경로에서 실행

```
# -d 는 백그라운드 실행
docker-compose up -d

# --build를 추가하면 기존이 삭제되어서 안됨
docker-compose up --build
```

- docker-compose 파일이 실행되면 jenkins 컨테이너와 nginx 컨테이너가 up 되어야 함

```
# docker 컨테이너 확인
docker ps -a

# docker 컨테이너 로그 확인
docker logs [컨테이너]

# doker 컨테이너 삭제 (컨테이너 삭제 후 이미지 삭제)
docker rm -f $(docker ps -qa)
```

```
# docker 이미지 삭제
docker image rm -f $(docker image ls -q)
```

3. SSL 인증서 발급 및 적용

init-letsencrypt 생성

- /home/ssafy/readed 경로에 init-letsencrypt.sh 생성

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains="i9a507.p.ssafy.io"
rsa_key_size=4096
data_path="./data/certbot"
email="davidpoplar@naver.com"
staging=0 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and replace existing certificate?
 (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi


if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ ! -e "$data_path/conf/ssl-dhpara
ms.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot_n
ginx/_internal/tls_configs/options-ssl-nginx.conf > "$data_path/conf/options-ssl-nginx.con
f"
  curl -s https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl-dhp
arams.pem > "$data_path/conf/ssl-dhparams.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
```

```
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days 1\
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
    -subj '/CN=localhost'" certbot
echo


echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo


echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

init-letsencrypt 실행

```
# 파일 실행 권한 변경
chmod +x init-letsencrypt.sh

# init-letsencrypt 쉘스크립트 실행
sudo ./init-letsencrypt.sh
```

- chmod +x : change mode 권한 변경 파일 실행 (x = execute)
- init-letsencrypt 실행 과정에서 nginx가 restart 된다.

nginx 재실행

- 인증서 위치에 맞게 docker 볼륨 마운트
- nginx.conf 인증키 주석 없이 docker-compose로 재실행

```
# nginx 컨테이너 스탑
docker-compose stop nginx

# nginx 컨테이너 삭제
docker rm readed_nginx_1

# 도커 컴포즈 다시 실행
docker-compose up -d
```

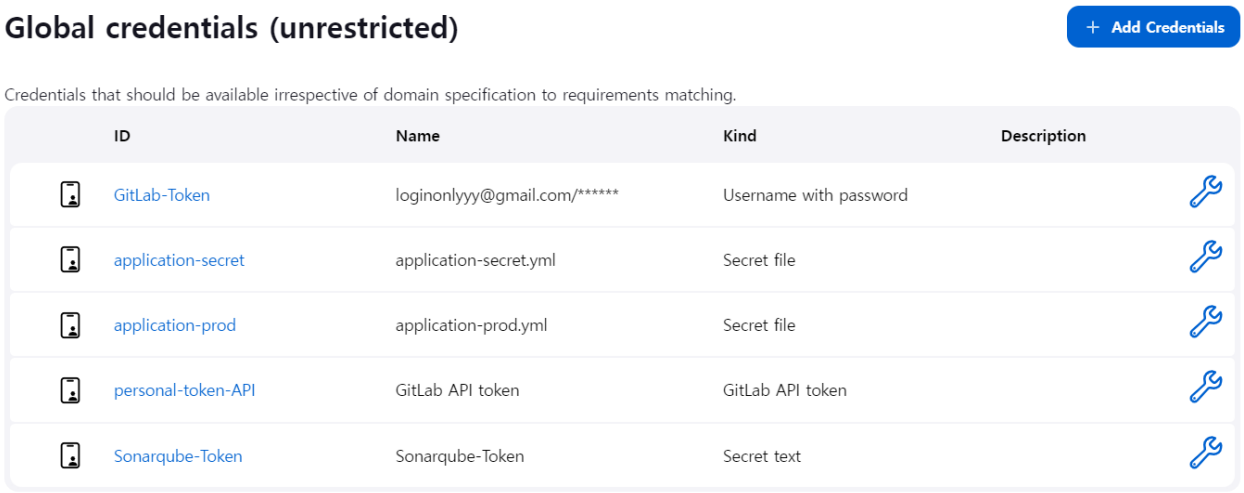
4. CI/CD 구축

Jenkins 환경설정

- 플러그인 설치
  - Gradle
  - Docker
  - Docker-compose
  - Sonarqube

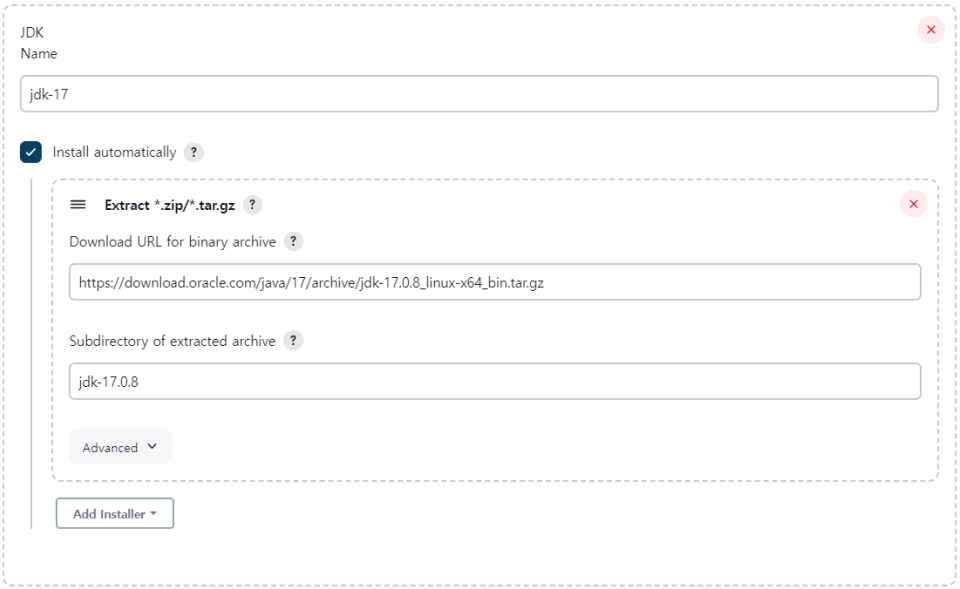
Credentials 설정

- 토큰
  - Gitlab-Token
  - Sonarqube-Token
- 주입 파일
  - application-secret.yml
  - application-prod.yml

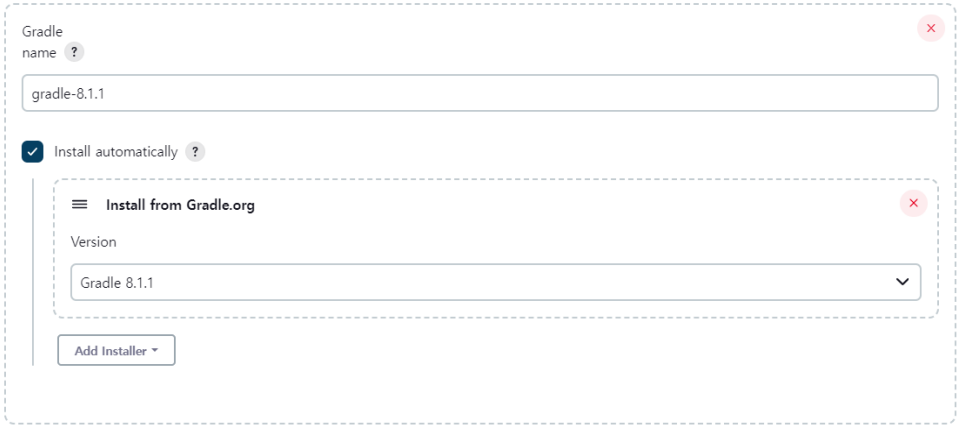


Tools 설정

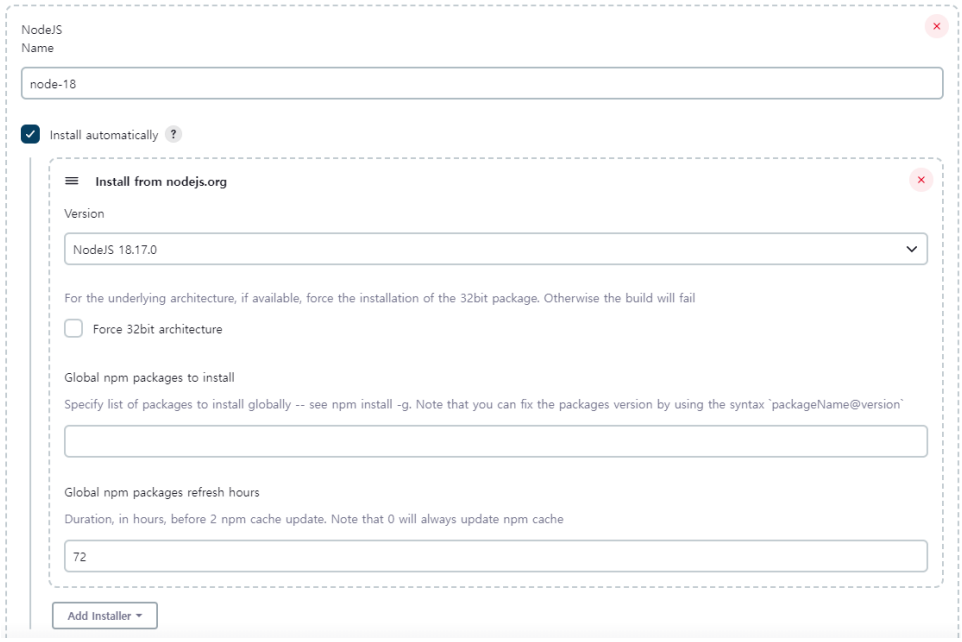
- JDK
  - https://download.oracle.com/java/17/archive/jdk-17.0.8_linux-x64_bin.tar.gz



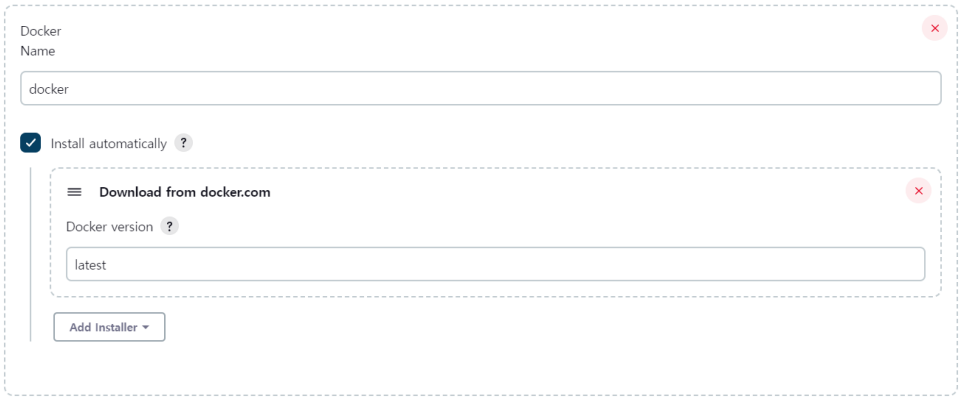
- Gradle



- NodeJS



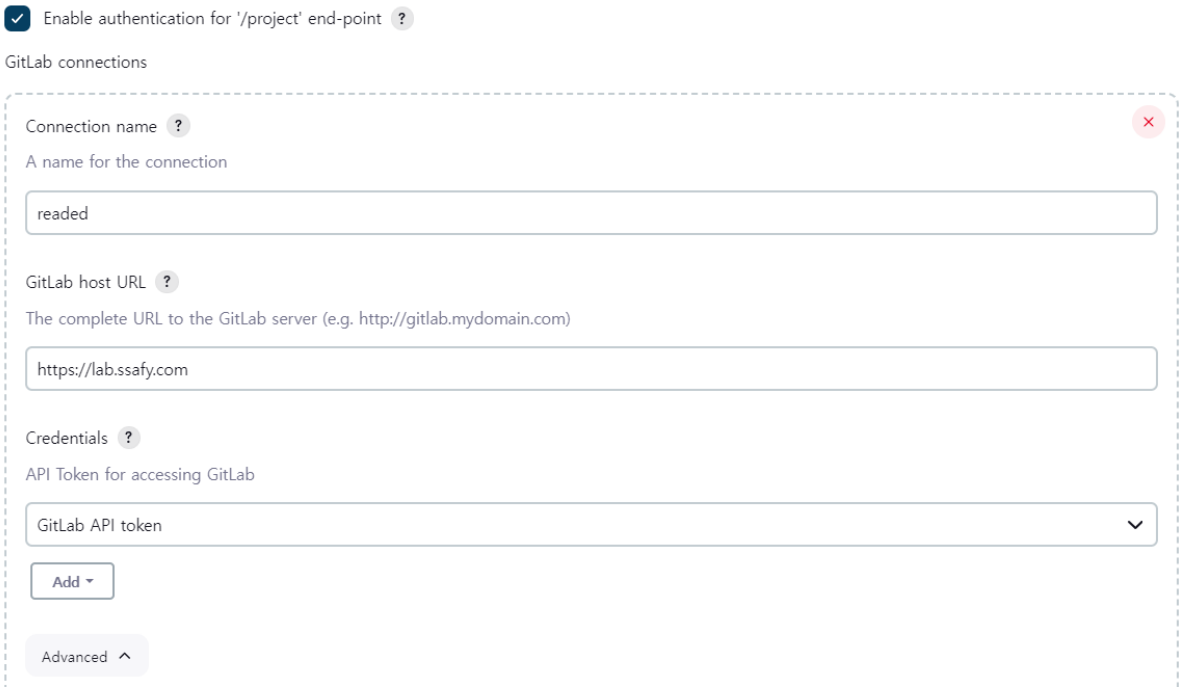
- Docker



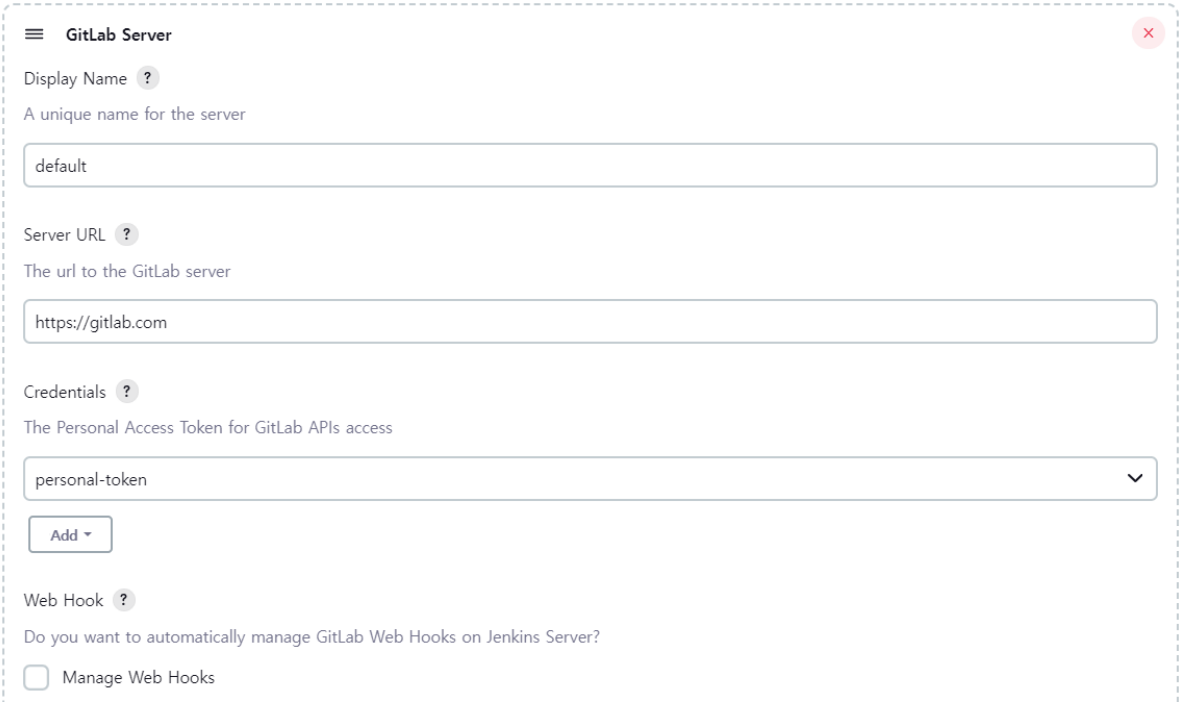
System 설정

- SonarQube servers

- Gitlab



- Gitlab

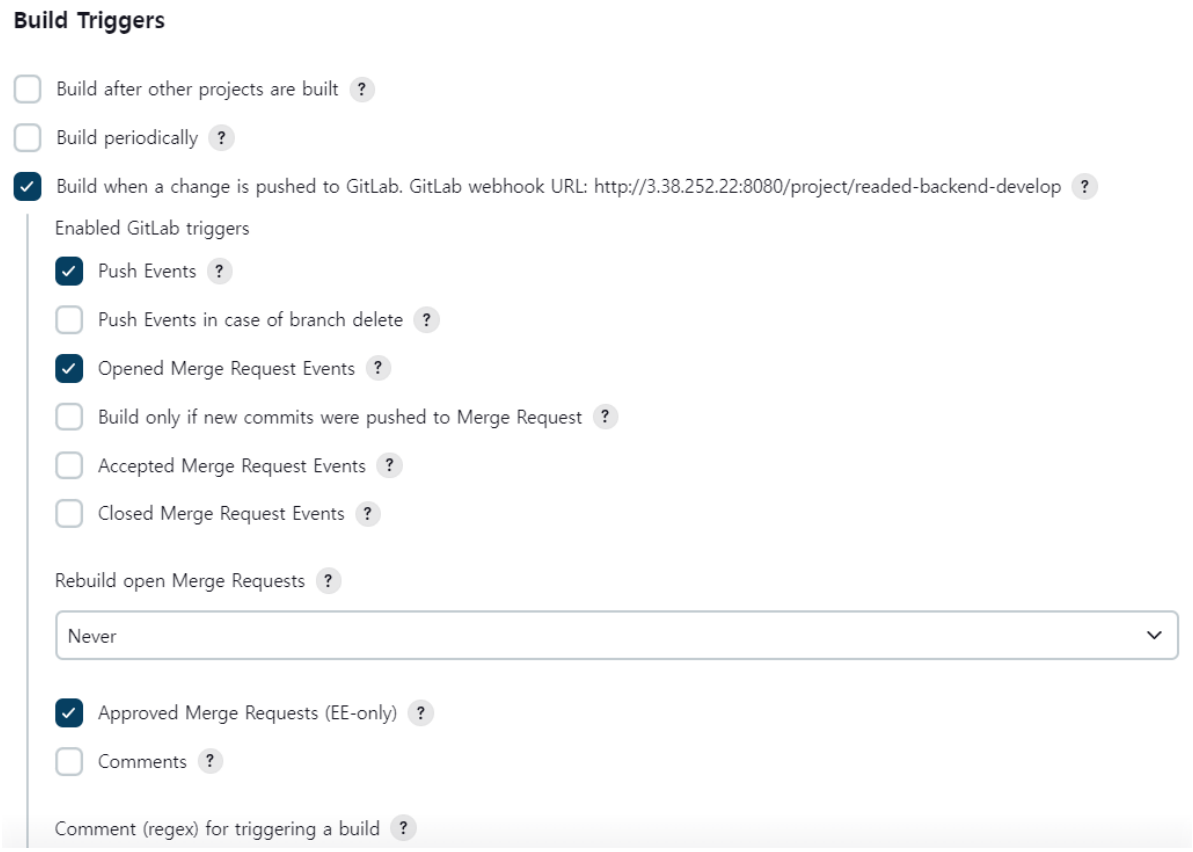


Jenkins Pipeline 생성

Build Triggers 설정

- webhook URL 저장 : http://3.38.252.22:8080/project/readed-backend-develop
- Advanced에서 Secret token generate 후 저장

▼ 캡쳐

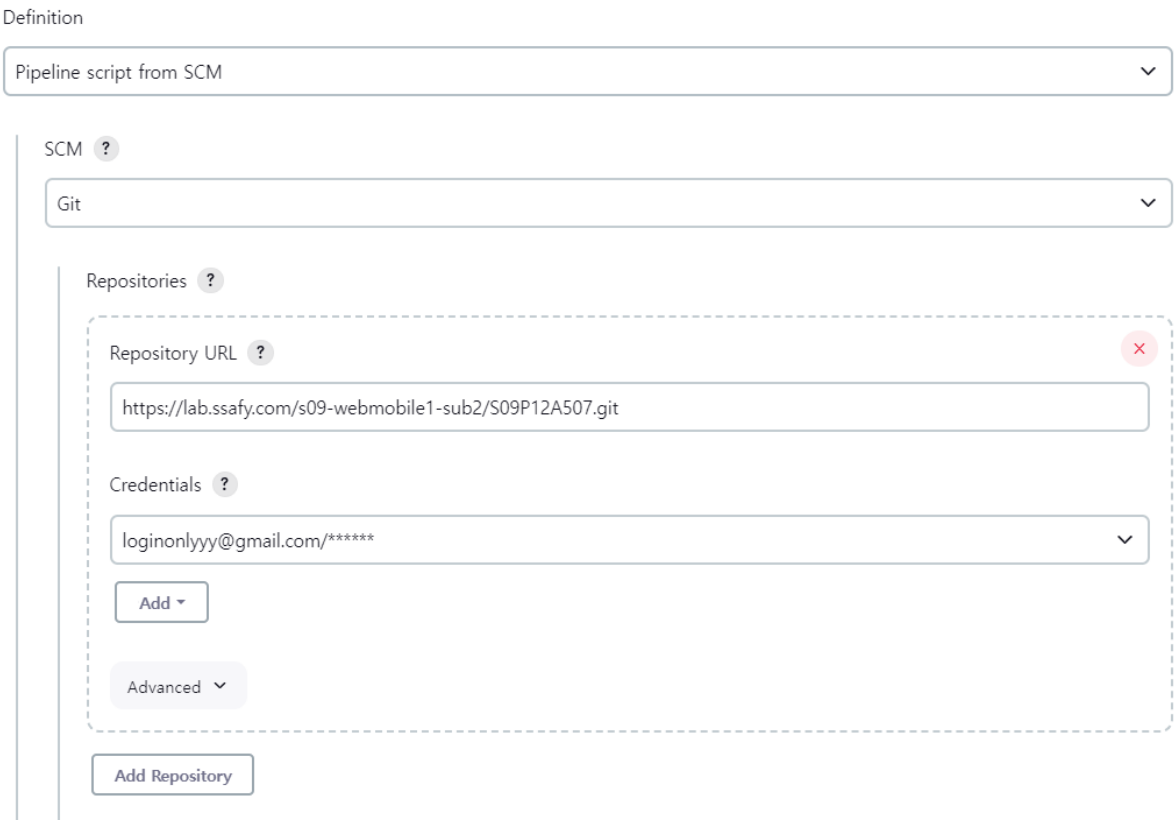

Build Triggers
Build after other projects are built
Build periodically
Build when a change is pushed to GitLab. GitLab webhook URL: http://3.38.252.22:8080/project/readed-backend-develop
Enabled GitLab triggers
Push Events
Push Events in case of branch delete
Opened Merge Request Events
Build only if new commits were pushed to Merge Request
Accepted Merge Request Events
Closed Merge Request Events
Rebuild open Merge Requests
Never
Approved Merge Requests (EE-only)
Comments
Comment (regex) for triggering a build

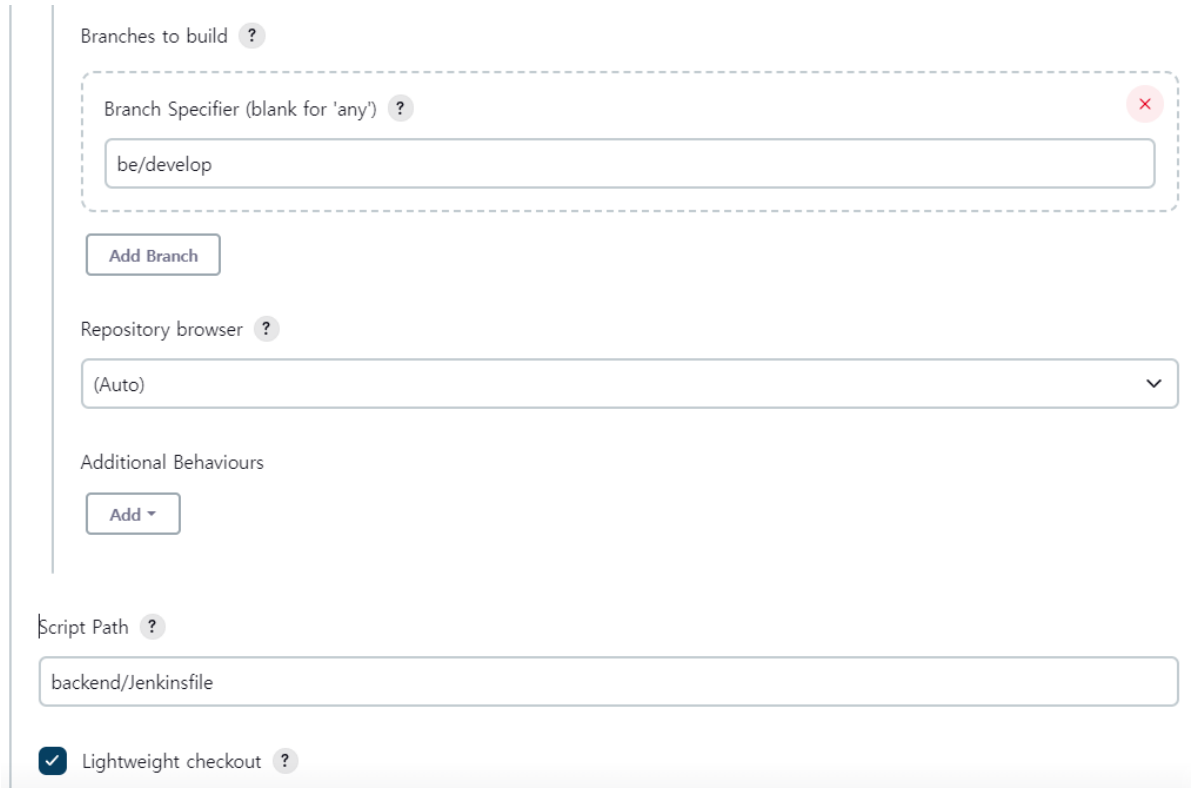
Pipeline 설정

- Pipeline script from SCM으로 설정
- Repository URL 입력
- Credentials 선택

- Branches to build 입력 : 빌드할 브랜치명
- Script Path 입력 : Jenkinsfile 위치

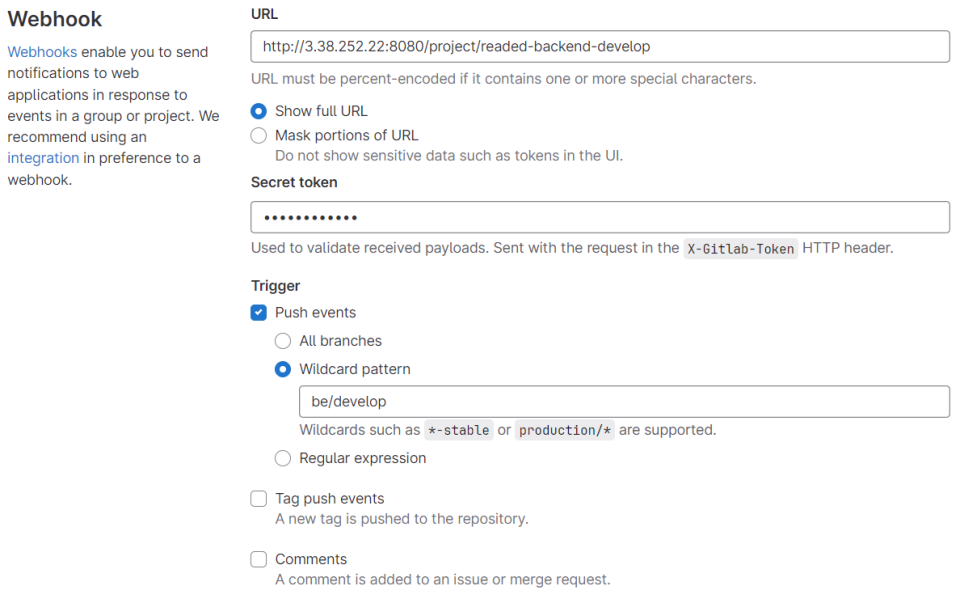
▼ 캡쳐





GitLab Webhook 생성

- URL : 젠킨스 파이프라인 webhook URL 입력
- Secret token : 젠킨스 파이프라인 generate한 Secret token 입력
- Trigger - Push events : trigger 시킬 브랜치명 설정



- 생길 수 있는 에러
  - 403 / 401
  - Secret token을 제대로 가져오지 않았거나 / URL 입력을 잘못했거나
- Test로 Webhook 테스트 가능

Jenkinsfile, Dockerfile 작성

backend Jenkinsfile

```
pipeline {

    agent any

    tools {
        gradle 'gradle-8.1.1'
        jdk 'jdk-17'
        dockerTool 'docker'
    }

    stages {
        stage('Clear current directory') {
            steps {
                sh'''
                    rm -rf *
```

```
                '''
            }
        }


        stage('Pull from GitLab') {
            steps {
                git url: 'https://lab.ssafy.com/s09-webmobile1-sub2/S09P12A507.git',
                    branch: 'be/develop',
                    credentialsId: 'GitLab-Token'
            }
        }

        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-secret', variable: 'secr
etFile'),
                                 file(credentialsId: 'application-prod', variable: 'prodFil
e')]) {
                    script {
                        sh 'cp $secretFile backend/src/main/resources/application-secret.y
ml'
                        sh 'cp $prodFile backend/src/main/resources/application-prod.yml'
                    }
                }
            }
        }

        stage('Build Backend') {
            steps {
                dir('backend') {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SSAFY-SonarQube') {
                    dir('backend') {
                        sh './gradlew sonarqube --stacktrace'
                    }
                }
            }
        }

        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect readed_server >/dev/null 2>&1; then
```

```
                        echo "container exists locally"
                        docker stop readed_server
                        docker rm readed_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect server >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }

        stage('Build and Deploy Docker') {
            steps {
                dir('backend') {
                    sh'''
                        echo [BE] Build Docker Image!
                        docker build -t server .
                        echo [BE] Run Docker Container!
                        docker run -dp 8081:8081 --name readed_server server
                    '''
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: tru
e).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tru
e).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', re
turnStdout: true).trim()
                mattermostSend(color: 'good', message: "✅ 빌드 & 배포 성공: ${env.JOB_NAME}
(<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n브랜치: be/develop\n커밋 메시지: ${GIT_COMMIT_MSG}
by ${Author_ID}(${Author_Name})\n<https://sonarqube.ssafy.com/dashboard?id=S09P12A507|Sona
rQube 분석 바로가기>")
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: tru
e).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tru
e).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', re
turnStdout: true).trim()
                mattermostSend(color: 'danger', message: "❌ 빌드 & 배포 실패: ${env.JOB_NAM
```

```
E} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n브랜치: be/develop\n커밋 메시지: ${GIT_COMMIT_M
SG} by ${Author_ID}(${Author_Name})\n<https://sonarqube.ssafy.com/dashboard?id=S09P12A507|
SonarQube 분석 바로가기>")
            }
        }
    }
}
```

backend Dockerfile

```
FROM openjdk:17-alpine
ARG JAR_FILE=build/libs/*-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar", "-Dspring.profiles.active=production"]
# changeset backend test19
```

frontend Jenkinsfile

```
pipeline {

    agent any

    tools {
        nodejs 'node-18'
        dockerTool 'docker'
    }


    stages {
        stage('Clear current directory') {
            steps {
                sh'''
                    rm -rf *
                '''
            }
        }

        stage('Pull from GitLab') {
            steps {
                git url: 'https://lab.ssafy.com/s09-webmobile1-sub2/S09P12A507.git',
                    branch: 'fe/develop',
                    credentialsId: 'GitLab-Token'
            }
        }

        stage('Build Frontend') {
            steps {
```

```
                dir('frontend') {
                    sh'''
                        npm install
                        npm run build
                    '''
                }
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SSAFY-SonarQube') {
                    dir('frontend') {
                        sh '''
                            npm run sonar
                        '''
                    }
                }
            }
        }

        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect readed_client >/dev/null 2>&1; then
                        echo "container exists locally"
                        docker stop readed_client
                        docker rm readed_client
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect client >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi client
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }

        stage('Build and Deploy Docker') {
            steps {
                dir('frontend') {
                    sh'''
                        echo [FE] Build Docker Image!
                        docker build -t client .
                        echo [FE] Run Docker Container!
                        docker run -dp 3000:3000 --name readed_client client
                    '''
                }
            }
        }
    }
```

```
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: tru
e).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tru
e).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', re
turnStdout: true).trim()
                mattermostSend(color: 'good', message: "✅ 빌드 & 배포 성공: ${env.JOB_NAME}
(<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n브랜치: fe/develop\n커밋 메시지: ${GIT_COMMIT_MSG}
by ${Author_ID}(${Author_Name})\n<https://sonarqube.ssafy.com/dashboard?id=S09P12A507|Sona
rQube 분석 바로가기>")
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: tru
e).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tru
e).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', re
turnStdout: true).trim()
                mattermostSend(color: 'danger', message: "❌ 빌드 & 배포 실패: ${env.JOB_NAM
E} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n브랜치: fe/develop\n커밋 메시지: ${GIT_COMMIT_M
SG} by ${Author_ID}(${Author_Name})\n<https://sonarqube.ssafy.com/dashboard?id=S09P12A507|
SonarQube 분석 바로가기>")
            }
        }
    }
}
```

frontend Dockerfile

```
# nginx 이미지 pull
FROM nginx
# app 디렉토리 생성
WORKDIR /app
# workdir에 build 폴더 생성 /app/build
RUN mkdir ./build
# build에서 build 폴더로 이동
ADD ./build ./build
# nginx의 기본 설정을 삭제
RUN rm -rf /etc/nginx/nginx.conf
# nginx 설정 파일 복사
COPY ./react-nginx.conf /etc/nginx/nginx.conf
# 80포트 오픈하고 nginx 실행
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

frontend react-nginx.conf

- 빌드된 프론트엔드 프로젝트를 nginx로 연결

```
user  nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    server {
        listen 3000;
        listen [::]:3000;

        server_name _;

        location / {
            root   /app/build;
            index  index.html index.htm;
            try_files $uri /index.html;
        }

        error_page   500 502 503 504  /50x.html;

        location = /50x.html {
            root   /usr/share/nginx/html;
        }
    }
}
```

5. OpenVidu 설정

기존 OpenVidu 삭제

```
# Linux Ubuntu 패키지 설치 리스트 확인
sudo apt list --installed
```

```
# 설치된 패키지 삭제
sudo apt remove openvidu

# /opt에서 openvidu 폴더 삭제
rm -r openvidu
(비어있는 디렉토리 제거 rmdir directoryname)
```

OpenVidu 설치

```
# Deploying OpenVidu CE on premises
cd /opt
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

.env 파일

- DOMAIN_OR_PUBLIC_IP=`i9a507.p.ssafy.io`
- OPENVIDU_SECRET=`MY_SECRET`
- CERTIFICATE_TYPE=`letsencrypt`
- LETSENCRYPT_EMAIL=`davidpoplar@naver.com`
- HTTP_PORT=`8442`
- HTTPS_PORT=`8443`

```
# OpenVidu configuration
# ----------------------
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvidu-config/

# NOTE: This file doesn't need to quote assignment values, like most shells do.
# All values are stored as-is, even if they contain spaces, so don't quote them.

# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=i9a507.p.ssafy.io

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

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
```

```
#                Users will see an ERROR when connected to web page.
# - owncert:     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=davidpoplar@naver.com

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

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

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

- 처음에 실행할 때는 포트 번호 바꾸지 않고 실행함
- 기본 80 443으로 ssl 인증을 받고 진행함
- 인증 후에는 처음의 8442와 8443 포트로 변경

docker-compose.override.yml

- SERVER_PORT=`5442`
- OPENVIDU_URL=`http://localhost:5443`

```
version: '3.1'

services:
    # ------------------------------------------------------------
    #
    #    Change this if your want use your own application.
    #    It's very important expose your application in port 5442
    #    and use the http protocol.
    #
    #    Default Application
    #
    #    Openvidu-Call Version: 2.28.0
    #
```

```
    # ------------------------------------------------------------
    app:
        image: openvidu/openvidu-call:2.28.0
        restart: on-failure
        network_mode: host
        environment:
            - SERVER_PORT=5442
            - OPENVIDU_URL=http://localhost:5443
            - OPENVIDU_SECRET=${OPENVIDU_SECRET}
            - CALL_OPENVIDU_CERTTYPE=${CERTIFICATE_TYPE}
            - CALL_PRIVATE_ACCESS=${CALL_PRIVATE_ACCESS:-}
            - CALL_USER=${CALL_USER:-}
            - CALL_SECRET=${CALL_SECRET:-}
            - CALL_ADMIN_SECRET=${CALL_ADMIN_SECRET:-}
            - CALL_RECORDING=${CALL_RECORDING:-}
        logging:
            options:
                max-size: "${DOCKER_LOGS_MAX_SIZE:-100M}"
```

frontend 포트번호

- BookclubMeeting

```
const OPENVIDU_SERVER_URL = `https://${window.location.hostname}:8443`;
```

backend 포트번호

- application-secret.yml

```
openvidu:
  secret: "MY_SECRET"
  url: "https://i9a507.p.ssafy.io:8443"
```

- BookclubServiceImpl

```
public BookclubServiceImpl(BookclubRepository bookclubRepository, BookRepository bookRepos
itory,
            MemberRepository memberRepository, ParticipantRepository participantRepositor
y, S3FileService s3FileService,
            @Value("${openvidu.url}") String url, @Value("${openvidu.secret}") String secr
et) {
```

OpenVidu 실행

```
# openvidu 처음 실행 시 80포트와 443포트를 사용하기 때문에 nginx보다 먼저 설치되어야
# nginx 중단
cd /home/ssafy/readed
docker ps -a
docker stop readed_nginx_1
docker rm readed_nginx_1

# openvidu 실행 (처음에 실행하면 인증이 진행되고 certificate 폴더가 생김)
cd /opt/openvidu
./openvidu start

# openvidu 멈추고 다시 실행할 경우
./openvidu stop

# nginx 재실행
cd /home/ssafy/readed
docker-compose up -d
```