CI/CD

Docker

▼ Docker 정보

버전 : 25.0.3

▼ 설치 과정

- 도커 설치 안내(공식 홈페이지)

1. 기존 설치되어 충돌이 발생할 수 있는 패키지 삭제

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

2. apt reprository 설정
   a. Add Docker's official GPG key : 리눅스 프로그램 설치 시 무결성 검증에 사용됨
   b. Add the repository to Apt sources :

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

   c. docker package 설치(최신 버전 설치 기준)

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

   d. 설치 확인 `sudo docker run hello-world`

```
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 // 도커 데몬은 이미지, 컨테이너, 네트워크, 볼륨과 같은 도커 객체를 관리하는 백그라운드 서비스
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

▼ Dockerfile - back

```
# 빌드 스테이지
FROM amazoncorretto:17.0.7-alpine AS builder
USER root
WORKDIR /back
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
# gradlew 실행 권한 부여
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# 실행 스테이지
FROM openjdk:17
WORKDIR /back
COPY --from=builder /back/build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
VOLUME /tmp
EXPOSE 8080
```

▼ Dockerfile - front

```
FROM node:lts-alpine as builder

WORKDIR /front-edu
```

```
ENV PATH /front-edu/node_modules/.bin:$PATH

COPY package.json /front-edu/package.json
COPY . .

RUN npm install
RUN npm install typescript @types/node @types/react @types/react-dom @types/jest react-router-dom redux react-redux

CMD ["npm", "start"]

EXPOSE 3000
```

Jenkins

▼ Jenkins 정보

버전 : 2.442

▼ 설치 과정

- 젠킨스 설치 안내 (https://hub.docker.com/_/jenkins)

0. 네트워크 설정

```
sudo docker network create jenkins-network
```

1. Docker Jenkins image download

```
sudo docker pull jenkins/jenkins
```

2. Docker Jenkins container 실행

```
sudo docker run -d -p 8080:8080 -p 50000:50000 jenkins/jenkins
```

3. 볼륨 설정

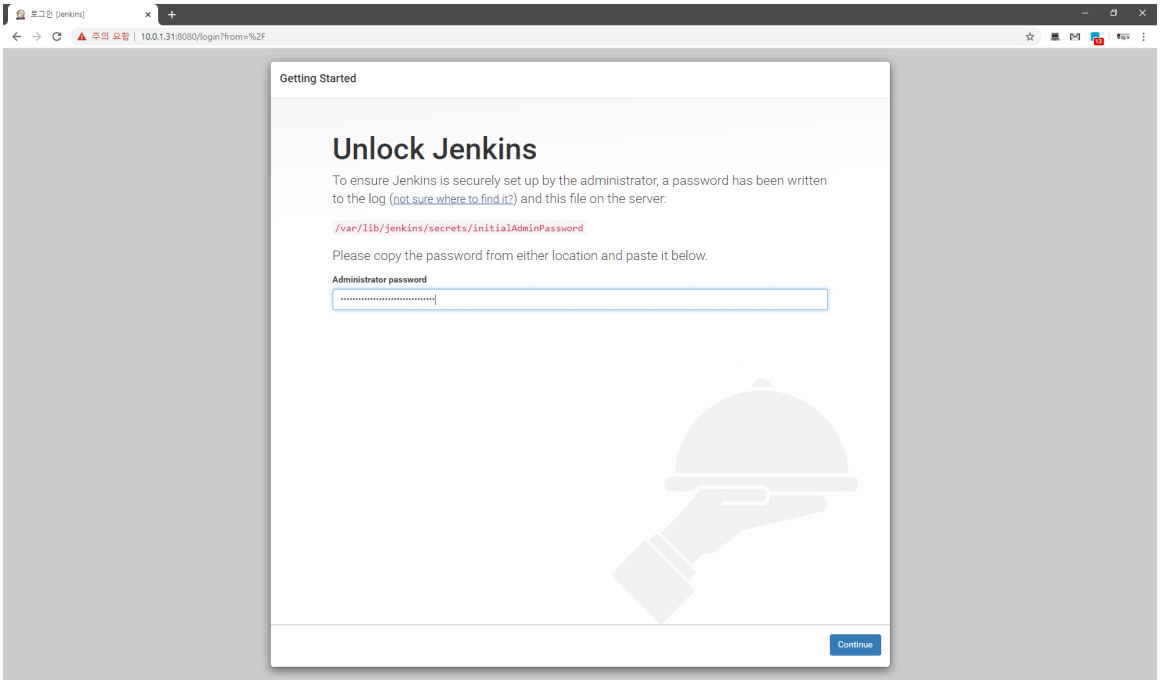
```
sudo docker run -p 8080:8080 -p 50000:50000 -v /your/home:/var/jenkins_home jenkins
```

4. jenkins Docker.sock 권한 오류(build 단계 fail)
   a. dial unix /var/run/docker.sock: connect: permission denied 와 같은 오류 메세지가 발생했을 때

```
sudo docker exec -it --user root jenkins bash
sudo chown root:docker /var/run/docker.sock
chown root:docker /var/run/docker.sock
```

▼ Jenkins 접속

1. http://{도메인주소}:{열어준 포트}/로 접속
2. 초기 비밀번호 화면



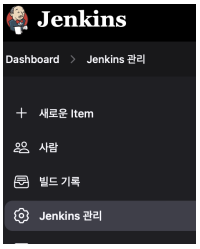
3. ec2 서버에서 초기 비밀번호 확인

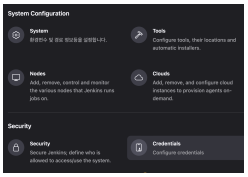
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

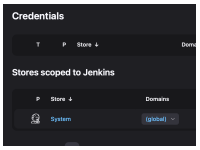
▼ jenkins-gitlab webhook 연결

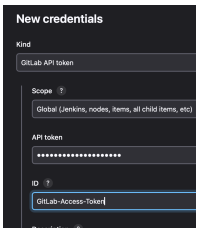
1. jenkins GitLab 플러그인 설치

▼ jenkins credential에 gitlab 토큰 등록 및 설정

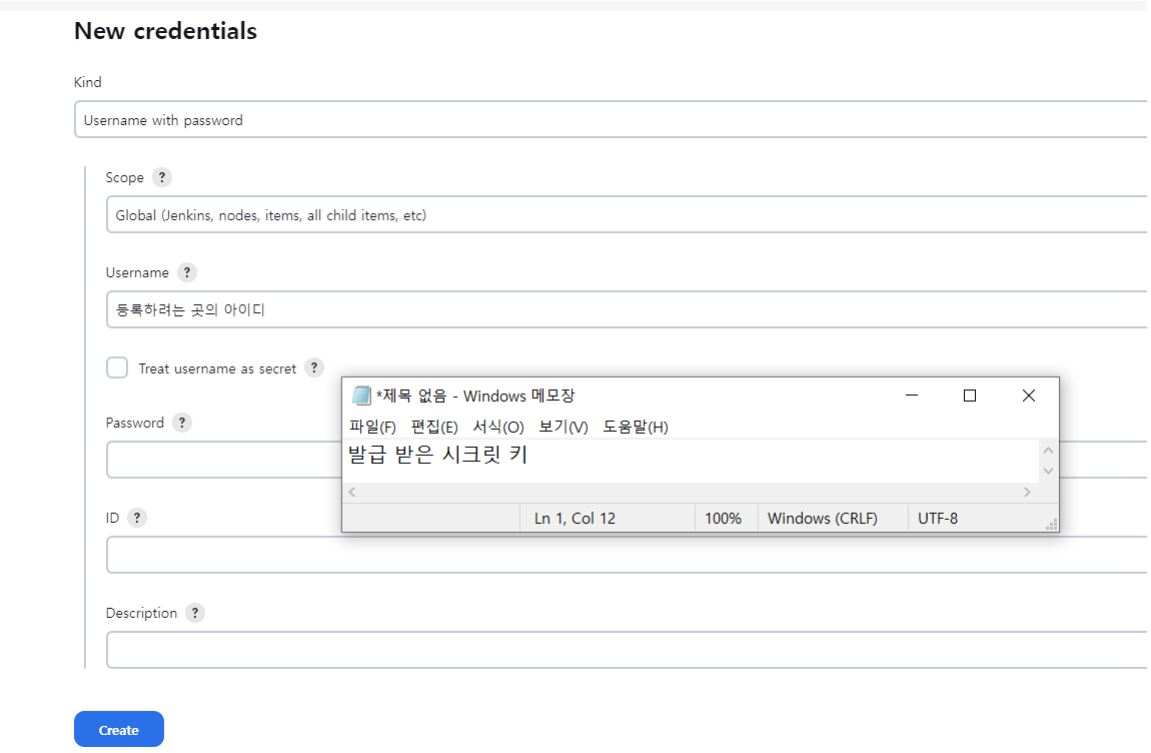




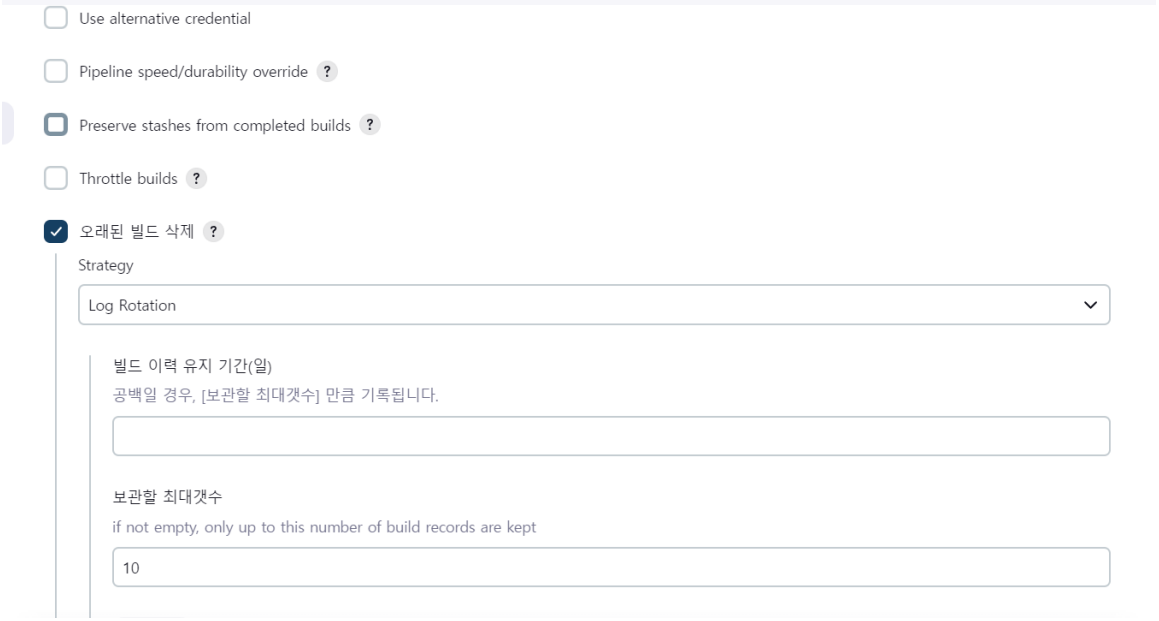




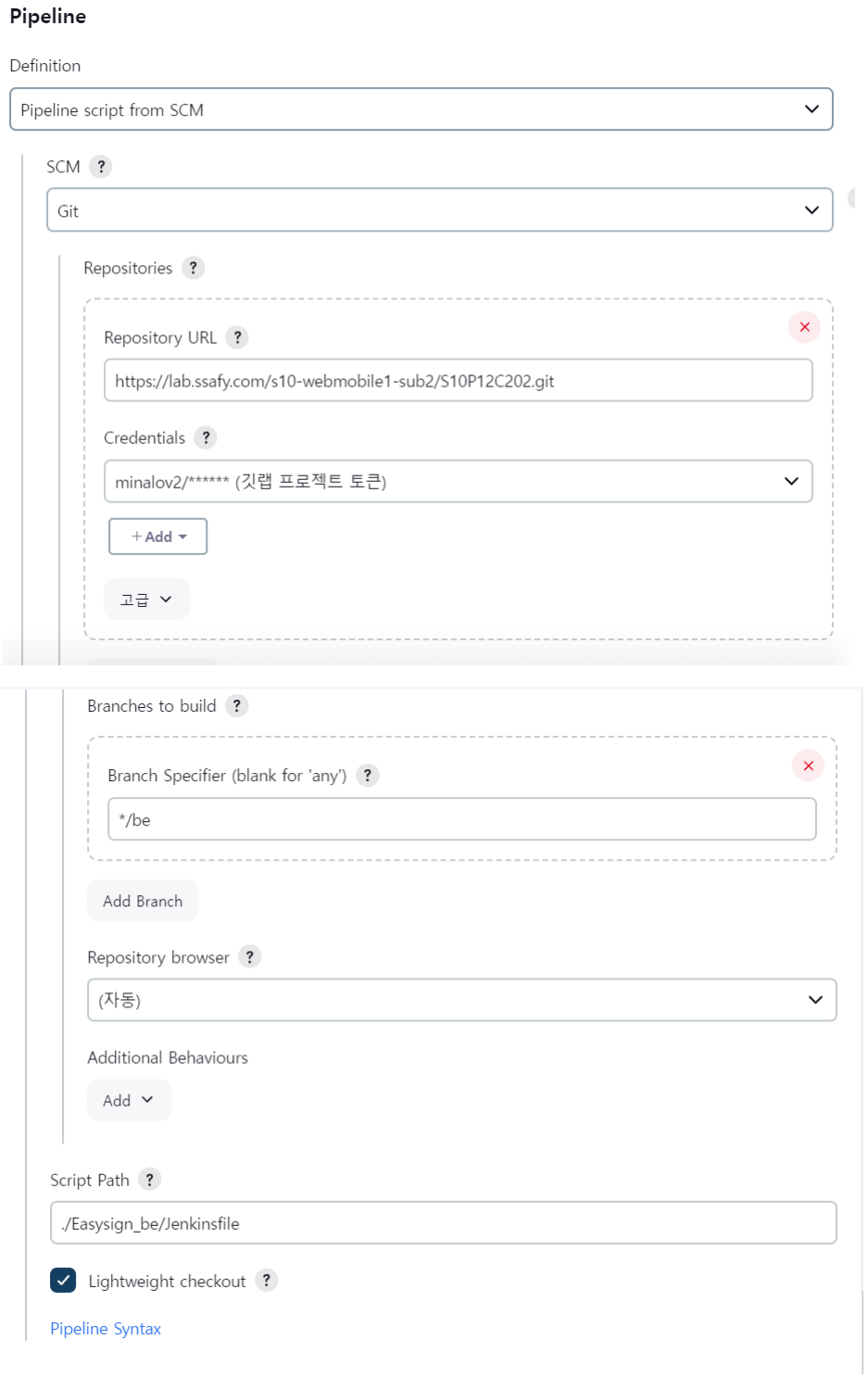
T	P	Store ↓	Domain	ID	Name
		System	(global)	GitLab-Project-access-Token	minalov2/****** (깃랩 프로젝트 토큰)
		System	(global)	GitLab-Access-Token2	minalov2/****** (개인access토큰)
		System	(global)	Docker-hub	doribari/****** (create)
		System	(global)	Github-access-token	dodokim98/****** (깃허브 토큰)



▼ pipeline 생성 및 설정

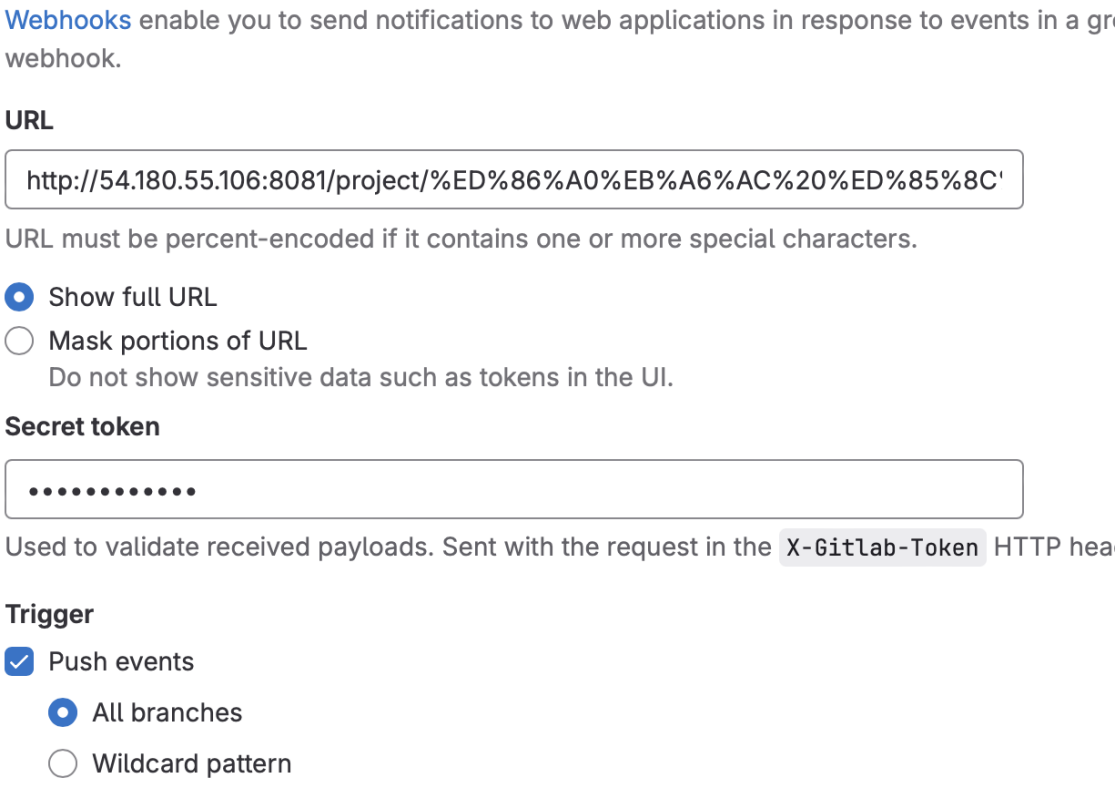


1. 고급 설정에서 secret token도 생성해준다.

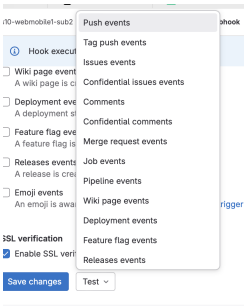


▼ gitlab webhook 등록

1. 프로젝트>설정>webhooks



2. 연결 테스트



▼ **Jenkinsfile - back**

```
pipeline {
    agent any
    environment {
        REPO = "s10-webmobile1-sub2/S10P12C202"
        DB_URL = "${env.DB_URL}"
        DB_USER = "${env.DB_USER}"
        DB_PASSWORD = "${env.DB_PASSWORD}"
        REDIS_HOST = "${env.REDIS_HOST}"
        REDIS_PASSWORD = "${env.REDIS_PASSWORD}"
        REDIS_PORT = "${env.REDIS_PORT}"
        MAIL_ID = "${env.MAIL_ID}"
        MAIL_PASSWORD = "${env.MAIL_PASSWORD}"
        MAIL_PORT = "${env.MAIL_PORT}"
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Setup Environment') {
            steps {
                dir("${env.WORKSPACE}/Easysign_be"){
                    script {
```

```
                    sh "ls -al"
                    sh "echo 'SUBMODULE CHECK'"
                    sh "ls ./src/main/resources"
                    sh "chmod +x ./gradlew"
                    sh "cat ./src/main/resources/application.yml"
                }
            }
        }
    }
    stage("Build") {
        steps {
            script {
                sh "ls -al"
                withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                    echo "도커허브 아이디: ${DOCKER_USER_ID}"
                    echo "도커허브 비밀번호: ${DOCKER_USER_PASSWORD}"
                    sh "docker build --no-cache -t ${DOCKER_USER_ID}/back Easysign_be"
                }
            }
        }
    }
    stage("Login") {
        steps {
            withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable:
                sh """
                    set +x
                    echo $DOCKER_USER_PASSWORD | docker login -u $DOCKER_USER_ID --password-stdin
                    set -x
                """
            }
        }
    }
    stage("Tag and Push") {
        steps {
            script {
                 withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariab
                    sh "docker push ${DOCKER_USER_ID}/back"
                 }
            }
        }
    }
    stage('Pull') {
        steps {
            script {
                withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                    sh "docker rmi doribari/back"     //images 날리기
                }
            }
        }
    }
    stage('Prune old images'){
        steps{
            script{
                sh "docker system prune --filter until=10h"
            }
        }
    }
    stage('Up') {
        steps {
            script {
                withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                    try{
                        sh "docker rm -f back || true"
                        sh "docker run -d --name back -p 8082:8080 \
                            -e DB_URL=${env.DB_URL} \
                            -e DB_USER=${env.DB_USER} \
                            -e DB_PASSWORD=${env.DB_PASSWORD} \
                            -e REDIS_HOST=${env.REDIS_HOST} \
                            -e REDIS_PASSWORD=${env.REDIS_PASSWORD} \
                            -e REDIS_PORT=${env.REDIS_PORT} \
                            -e MAIL_ID=${env.MAIL_ID} \
                            -e MAIL_PASSWORD=${env.MAIL_PASSWORD} \
                            -e MAIL_PORT=${env.MAIL_PORT} \
                            doribari/back"
                    } catch (Exception e){
                        sh "docker restart back || true"  // Ignore error if container doesn't exist
                    }
```

```
                    }
                }
            }
        }
    }
}
```

▼ Jenkinsfile - front

```
pipeline {
    agent any
    environment {
        REPO = "s10-webmobile1-sub2/S10P12C202"
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage("Build") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                        sh "docker build -t ${DOCKER_USER_ID}/front-edu Easysign_fe-edu"
                        sh "docker system prune --filter until=10h"
                    }
                }
            }
        }
        stage("Login") {
            steps {
                withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 
                    sh """
                        set +x
                        echo $DOCKER_USER_PASSWORD | docker login -u $DOCKER_USER_ID --password-stdin
                        set -x
                    """
                }
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                     withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariab
                        sh "docker push ${DOCKER_USER_ID}/front-edu"
                     }
                }
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker system prune --filter until=1h"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                        sh "docker pull ${DOCKER_USER_ID}/front-edu"
                    }
                }
            }
        }
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariabl
                        try{
                            sh "docker stop -f front-edu || true"
                            sh "docker rm -f front-edu || true"
                            sh "docker run -d --name front-edu -p 8084:3000 ${DOCKER_USER_ID}/front-edu"
                        } catch (Exception e){
                            sh "docker restart front-edu"
                        }
```

```
                    }
                }
            }
        }
    }
}
```

Nginx

▼ Nginx 정보

버전 : 1.18.0 (Ubuntu

▼ Nginx 설정

1. 패키지 업데이트 및 업그레이드

```
- sudo apt update

- sudo apt upgrade or sudo add-apt-repository –remove ppa:certbot/certbot

- free -h (현재 메모리 용량 확인)
```

2. 방화벽 설정

```
- sudo ufw status (방화벽 허용)

- sudo ufw allow [포트번호]  (방화벽 허용할 포트번호 입력)
```

3. nginx 설치

```
- sudo apt install nginx -y

- sudo systemctl status nginx ( 설치 후 상태 확인)
```

4. SSL 설치

```
SSL 설치
```

5. Certbot 설치

```
- sudo apt-get install certbot python3-certbot-nginx

- sudo certbot -nginx (certbot nginx 연결)
```

```
[

1. 이메일 입력
2. 약관 동의 : Y
3. 이메일 수신 동의
4. 도메인 입력 : i10c204.p.ssafy.io
5. http 입력시 Redirect

]
```

6. Nginx 환경 설정

```
- sudo cd /etc/nginx/sites-available/{파일명}.conf
```

7. Nginx 설정 파일

```
server {
    server_name i10c202.p.ssafy.io;

    location / {
        return 301 https://easysign.shop$request_uri;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i10c202.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i10c202.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 {
    if ($host = i10c202.p.ssafy.io ) {
        return 301 https://easysign.shop$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name i10c202.p.ssafy.io;
    return 404; # managed by Certbot
}

server {
        server_name easysign.shop;

        location / {
                proxy_pass http://localhost:8083;
        }

        location /api {
                proxy_pass http://localhost:8082;
        }

        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/easysign.shop/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/easysign.shop/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 {
    if ($host = easysign.shop ) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name easysign.shop;
    return 404; # managed by Certbot
}


server {
        server_name edu.easysign.shop;

        location / {
                proxy_pass http://localhost:8084;
        }

        location /api {
                proxy_pass http://localhost:8082;
        }

        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/edu.easysign.shop/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/edu.easysign.shop/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 {
    if ($host = edu.easysign.shop ) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name edu.easysign.shop;
    return 404; # managed by Certbot
}

server {
    server_name jenkins.easysign.shop;

    location / {
        proxy_pass http://localhost:8081;
    }
```

```
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/jenkins.easysign.shop/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/jenkins.easysign.shop/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 {
    if ($host = jenkins.easysign.shop) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    server_name jenkins.easysign.shop;
    listen 80;
    return 404; # managed by Certbot
}
```

Redis

▼ Redis 정보

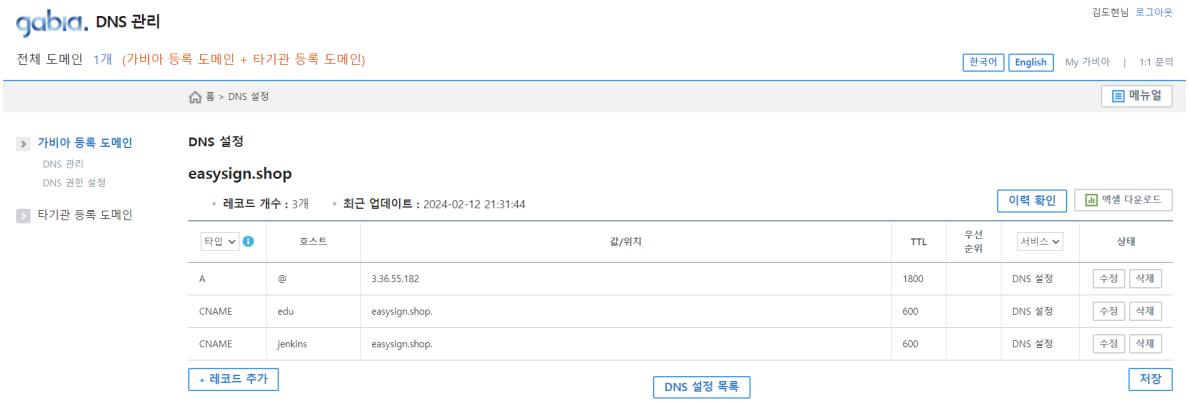
버전 : redis 7.2.4

▼ Redis 설치 및 설정

```
# Redis에 필요한 패키지를 먼저 설치한다.
sudo apt install lsb-release curl gpg

# Redis를 설치한다.
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb_release -cs) main"

sudo apt-get update
sudo apt-get install redis

# Redis의 설정을 변경하기 위해 redis.conf의 권한을 수정한다.
sudo chown root:root /etc/redis/redis.conf

# Redis의 설정을 변경한다.
sudo vi /etc/redis/redis.conf
    # requirepass 비밀번호
    # bind 접근 가능한 ip
    # port 포트번호

# 다시 권한을 원래대로 돌린다.
sudo chown redis:redis /etc/redis/redis.conf

# 바뀐 설정을 적용시킨다.
sudo systemctl restart redis-server.service

# 재부팅시에도 자동으로 실행되되록 한다.
sudo systemctl enable redis-server.service
```

가비아 도메인

▼ 도메인 설정



- A 타입 : @ 모든 url 허용 ex) edu.easysign.shop
- CNAME : 도메인에 앞에 붙는 것 ex) edu.easysign.shop