



CRIT 포팅 매뉴얼

1. 개발환경

1.1 Frontend

1.2 Backend

1.3 Server

1.4 Database

1.5 UI/UX

1.6 IDE

1.7 형상 / 이슈관리

1.8 기타 툴

2. 환경변수

3. EC2 세팅

3.1 Docker, Docker-compose Engine 설치

3.2 EC2 포트 정리

3.3 Nginx 설치 & reverse proxy 설정 & SSL인증서 발급

4. DB설정

5. 빌드 및 배포

5.1 Openvidu

5.2 Backend

5.3 Frontend

6. 외부서비스

6.1 소셜 로그인 - kakao

6.2 S3 & cloudfront

6.3 jasypt

6.4 sonarqube

6.5 mattermost

1. 개발환경

1.1 Frontend

- Node js 18.19.0
- React 18.2.0
- axios 1.4.0
- jquery 3.7.0
- swiper 10.1.0

1.2 Backend

- Java 11
 - java OpenJDK
 - Spring Boot

- Spring Data JPA
- Spring Security
- oauth2
- JUnit
- Lombok
- Swagger
- Gradle 1.0.10
- drewnoakes 2.18.0

1.3 Server

- Ubuntu 20.04.3
- Nginx 1.18.0
- Docker 24.0.4
- Docker-Compose 2.20.2
- OpenVidu 2.28.0
- Jenkins 2.401.3

1.4 Database

- mariaDB 2.4.1

1.5 UI/UX

- Figma

1.6 IDE

- Visual Studio Code
- IntelliJ IDEA

1.7 형상 / 이슈관리

- Gitlab
- Jira

1.8 기타 툴

- Sonarqube 4.6.2
- S3 2.2.6
- Cloudfront
- mattermost 7.8.6
- jasypt 2.14.2
- postman 10.17.0
- teachable machine 0.8.5

2. 환경변수

- Backend

```
SERVER_PORT
spring.datasource.url
spring.datasource.username
spring.datasource.password
spring.jpa.hibernate.ddl-auto
spring.security.oauth2.client.registration.google.client-id
spring.security.oauth2.client.registration.google.client-secret
```

```

spring.security.oauth2.client.registration.google.redirect-uri
spring.security.oauth2.client.registration.naver.client-id
spring.security.oauth2.client.registration.naver.client-secret
spring.security.oauth2.client.registration.naver.redirect-uri
spring.security.oauth2.client.registration.kakao.client-id
spring.security.oauth2.client.registration.kakao.client-secret
spring.security.oauth2.client.registration.kakao.redirect-uri
OPENVIDU_URL
OPENVIDU_SECRET
app.auth.token-secret
cloud.aws.s3.bucket
cloud.aws.credentials.access-key
cloud.aws.credentials.secret-key
cloud.aws.region.static
cloud.aws.cloudfront.url

```

3. EC2 세팅

3.1 Docker, Docker-compose Engine 설치

```

Docker install
Sudo apt-get update
sudo apt-get install docker -y

Docker-compose install
sudo apt install docker-compose
sudo chmod +x /home/ubuntu/docker-compose.yml

```

3.2 EC2 포트 정리

Port번호	내용
22	SSH
80	HTTP(HTTPS로 redirect)
443	HTTPS
3000	Nginx, React
3478	Openvidu
8080	Sprung Boot(Docker)
8081	Sonarqube(Docker)
9000	Jeknins(Docker)

3.3 Nginx 설치 & reverse proxy 설정 & SSL인증서 발급

```

# 1. Nginx 설치
sudo apt-get install nginx
nginx -v
# 2. Let's Encrypt 설치 및 SSL 발급
sudo apt-get install letsencrypt
sudo systemctl stop nginx
sudo letsencrypt certonly --standalone -d 도메인명
# 3. Nginx 설정파일 생성
cd /etc/nginx/sites-available
vi default
#####
##설정파일 내용
server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    ssl_certificate /etc/letsencrypt/live/i9d201.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i9d201.p.ssafy.io/privkey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!MD5;
    root /home/ubuntu/crit_fe/build;

    index index.html index.htm index.nginx-debian.html;
    client_max_body_size 300M;
    server_name _;

    location / {
        try_files $uri /index.html;
    }
}

```

```

# api reverse proxy
location /api/ {
    proxy_pass http://localhost:8080/;
}

server {
    listen 80;
    listen [::]:80;

    server_name i9d201.p.ssafy.io;

    return 301 https://i9d201.p.ssafy.io;
}
# 4. nvm, nodejs 설치
sudo apt-get install -y curl
sudo apt update
sudo apt install nodejs
nodejs -v
sudo apt install npm

```

4.DB설정

- ssafy에서 제공해주는 mariaDB사용

```

spring:
  servlet:
    multipart:
      max-file-size: 300MB
      max-request-size: 300MB
  datasource:
    url: jdbc:mariadb://stg-yswa-kr-practice-db-master.mariadb.database.azure.com:3306/S09P13D201?serverTimezone=UTC&useUnicode=true&c
    username: S09P13D201
    password: sMTWKgddDr
    driver-class-name: org.mariadb.jdbc.Driver
    hikari:
      maximum-pool-size:3
  main:
    allow-circular-references: true

  profiles:
    include: db

  jpa:
    hibernate:
      ddl-auto: update #create update none
    # show-sql: true
    properties:
      hibernate:
        # show_sql: true
        # format_sql: true

```

5.빌드 및 배포

5.1 Openvidu

```

#1 openvidu를 배포하기 : root 권한을 얻어야 한다.
sudo su

#2 openvidu를 설치하기 위해 권장되는 경로인 /opt로 이동
cd /opt

#3 openvidu 설치
curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh> | bash

#4 설치 후 openvidu가 설치된 경로로 이동
$ nano .env

# OpenVidu configuration
# -----
# 사용 도메인 주소 입력
DOMAIN_OR_PUBLIC_IP=

# 오픈비두 서버와 통신을 위한 시크릿 키 입력 (어플리케이션 서버와 통신할 때 필요)
OPENVIDU_SECRET=

# Certificate type
CERTIFICATE_TYPE=letsencrypt

```

```
# 인증서 타입이 letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL=user@example.com

# HTTP port
HTTP_PORT=8442

# HTTPS port(해당 포트를 통해 오픈비두 서버와 연결)
HTTPS_PORT=8443

#5 도메인 또는 public IP와 openvidu와 통신을 위한 환경설정
$ ./openvidu start

Creating openvidu-docker-compose_coturn_1      ... done
Creating openvidu-docker-compose_app_1         ... done
Creating openvidu-docker-compose_kms_1         ... done
Creating openvidu-docker-compose_nginx_1       ... done
Creating openvidu-docker-compose_redis_1       ... done
Creating openvidu-docker-compose_openvidu-server_1 ... done

-----

OpenVidu Platform is ready!
-----

* OpenVidu Server: https://DOMAIN_OR_PUBLIC_IP/

* OpenVidu Dashboard: https://DOMAIN_OR_PUBLIC_IP/dashboard/

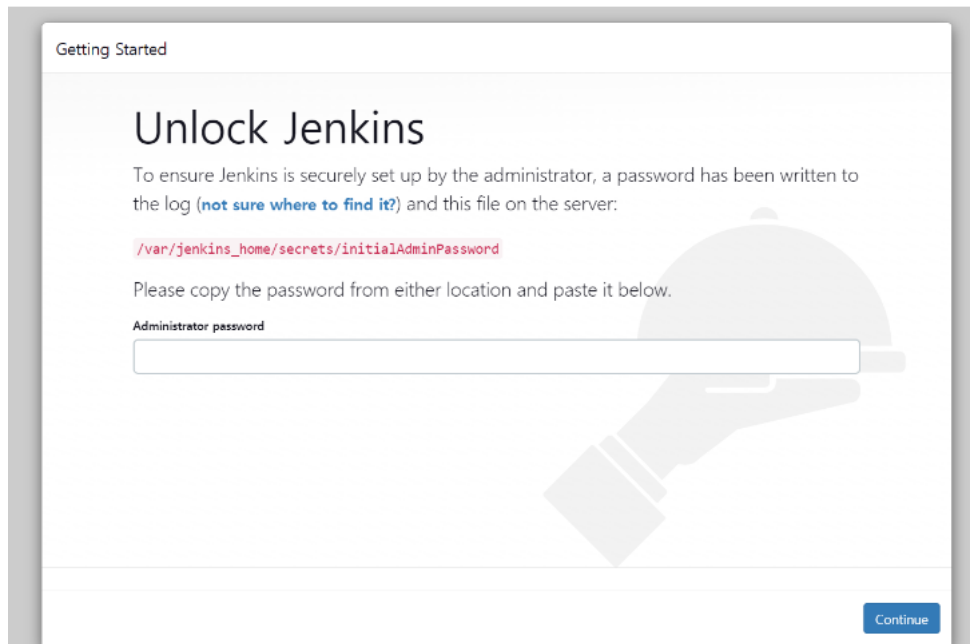
-----
```

5.2 Backend

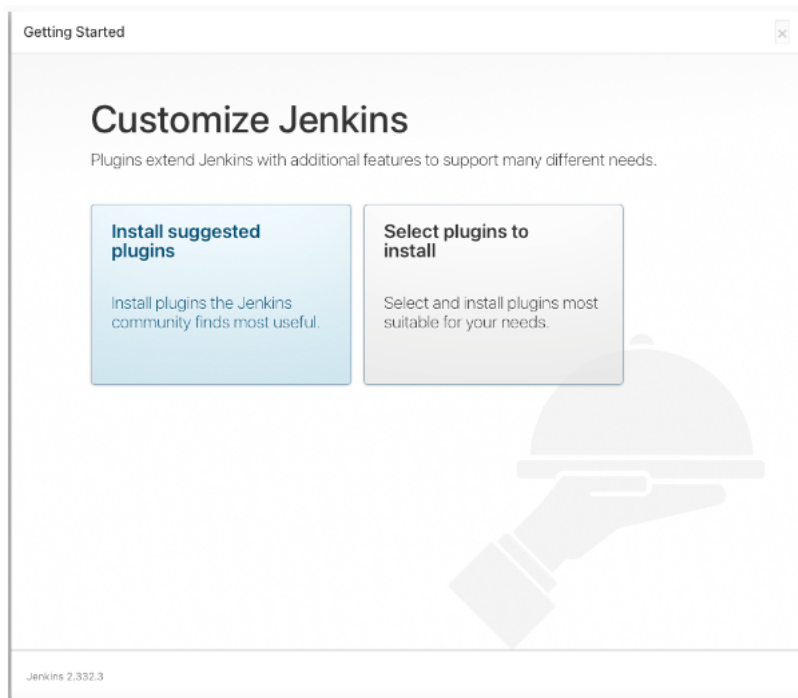
- docker-compose.yml파일을 만들고 여러개의 컨테이너를 실행할 수 있는 코드를 구현한다.
위치는 /home/ubuntu/에 만들었다.
 - sudo vim docker-compose.yml파일을 만들고 아래의 코드를 입력한다. 포트는 사전에 열어둬야한다.

```
version: '3'
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - 9000:8080
    privileged: true
    user: root
```

- /var/run/docker.sock를 등록을 해야 jenkins 안에서도 docker를 실행할 수 있다.
- **volumes** : 도커 컨테이너의 데이터는 컨테이너가 종료되면 휘발된다. 도커 컨테이너의 데이터를 보존하기 위해 사용한다.
(/var/jenkins_home 이라는 디렉터리를 /jenkins 와 마운트하고 데이터를 보존할 수 있다.)
- docker-compose.yml이 존재하는 경로에서 sudo docker-compose up -d를 입력하여 실행한다. -d는 백그라운드에서 실행하겠다는 명령어다.
- docker ps 를 입력하여 컨테이너가 실행 되는지 확인한다. 만약 안보이면 docker ps -a로 확인한다. -a 명령어는 실행되지 않는 것도 볼 수 있는 명령어다. 확인후 도메인과 포트번호를 입력하여 들어간다.
- 입력한 도메인과 포트를 입력해 jenkins를 실행하고 다음 화면과 같이 나오며 성공이다.



- docker logs jenkins(name이다) 을 통해 비밀번호를 확인하고 빈칸에 넣는다.
- 젠킨스 플러그인을 전부 설치한다.



- 계정을 생성한다.

Getting Started

Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 2.332.3

[Skip and continue as admin](#) [Save and Continue](#)

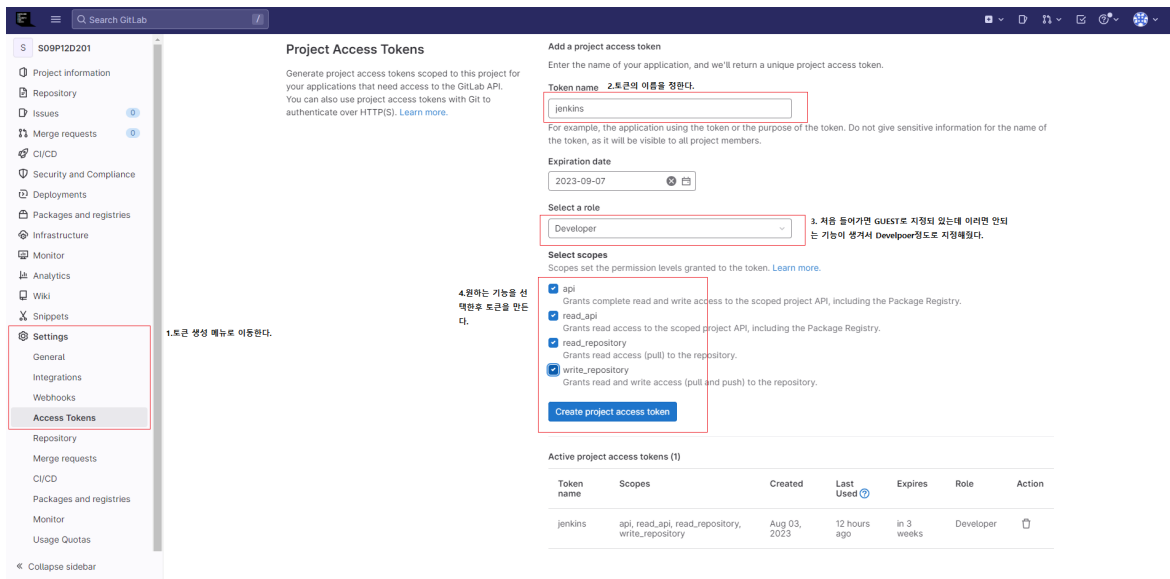
- 다시 서버로 돌아가서 jenkins 도커 컨테이너 안으로 들어간다.

```
sudo docker exec -it jenkins /bin/bash
```

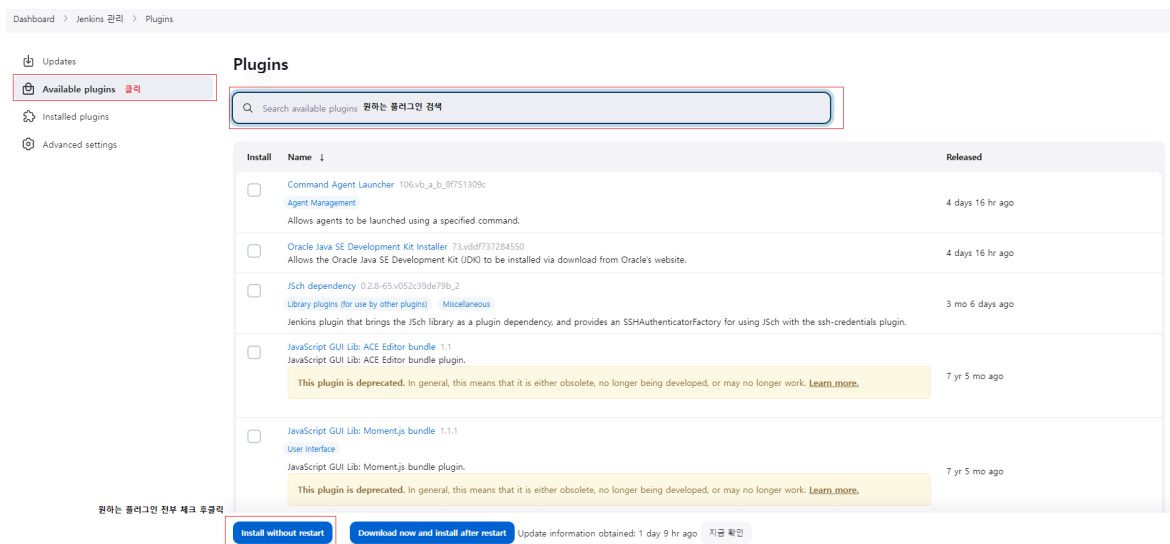
- 컨테이너 안에 들어가면 docker를 설치한다. 둘중 하나의 코드를 입력해 실행 후 확인 하기위해 안에서 docker ps를 입력한다.

```
docker exec -it ubuntu_jenkins_1 /bin/bash : container 접속
curl -fsSL https://get.docker.com -o get-docker.sh : docker 설치
sh get-docker.sh
혹은
apt-get update && \
apt-get -y install apt-transport-https \
ca-certificates \
curl \
gnupg2 \
software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
$(lsb_release -cs) \
stable" && \
apt-get update && \
apt-get -y install docker-ce
```

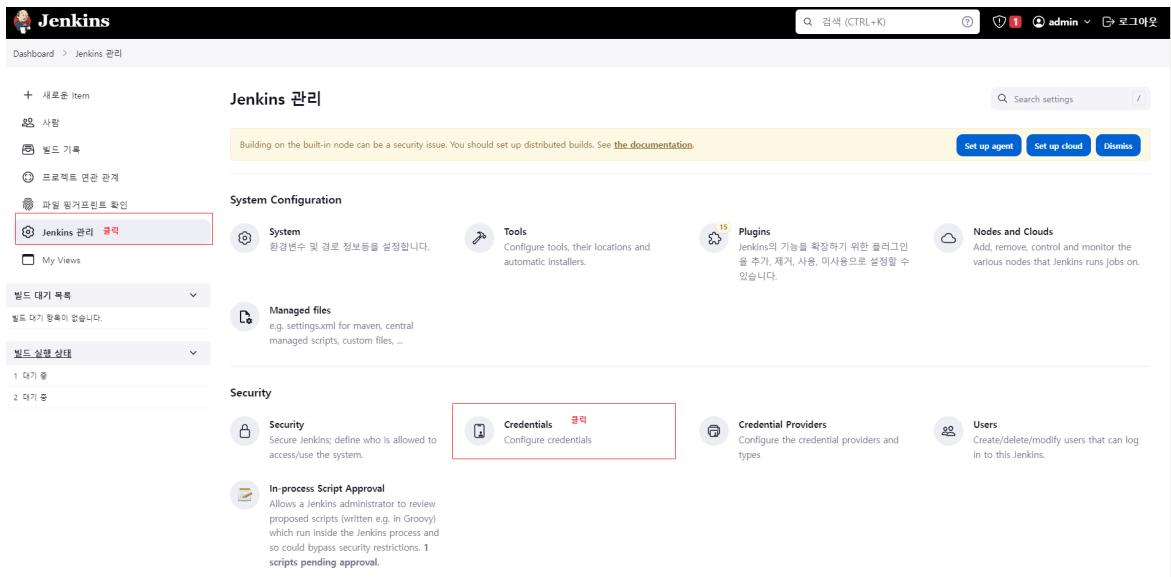
- gitlab을 사용해서 gitlab을 연결하기 위한 토큰을 생성을 한다.



- plugins에 들어가서 available plubins에 들어가 gitlab과 ssh agent, docker를 설치한다.



- 젠킨스에 접속해서 로그인 후 jenkins관리로 이동 후 Credentials를 클릭한다.



- 새로운 Credentials를 만들어준다.

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	jenkins	jenkins/*****
		System	(global)	ubuntu	ubuntu

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

- 발급받은 토큰을 jenkins에 등록을 해준다.

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ? 1.발급한 토큰의 이름을 입력한다.

☐ Treat username as secret ?

Password ? 2.발급한 토큰의 키값을 입력한다.

ID ? 3.등록하고 싶은 이름을 입력한다. 혹은 발급한 토큰의 이름과 동일하게 입력해도 된다.

Description ?

Create

- 이번엔 ssh에 연결할 수 있는 키를 등록한다.

Kind 1.그림과 동일한 카테고리로 변경한다.

SSH Username with private key

Scope ?

Global (Denkins, nodes, items, all child items, etc)

ID ? 2. 등록하고 싶은 ID를 입력한다.

Description ?

3. EC2 사용자 이름을 등록하는데 끝자는 ubuntu

Username 를 사용해서 ubuntu라 입력했다.

☐ Treat username as secret ?

Private Key 4.체크를 해서 EC2에 접속 할 수 있는 ppm 파일의 내용을 복사 붙여넣기 한다.

☒ Enter directly

Key +).pem파일은 백오점으로 열면 안의 내용이 나오니 전부 복사해 붙여 넣는다.

Enter New Secret Below

Create

- 이제 메인 페이지에서 새로운 item을 클릭하고 FreeStyleProject를 선택 후 원하는 이름을 설정 후 생성한다.
 - 생성 후 바로 입력해도 되고 나가서 언제든지 해당 item의 구성을 클릭하여 수정할 수 있다.
- 구성의 소스코드 관리로 이동하여 git을 클릭하고 git 주소, 등록된 git토큰 배포할 브랜치를 선택한다.

Configure

General

소스 코드 관리

소스 코드 관리

None

Git ? 클릭

Repositories ?

Repository URL ? 1.배포하고자 하는 git주소를 입력한다.

https://lab.ssafty.com/s09-webmobile1-sub2/S09P12D201.git

Credentials ? 2.jenkins에 등록된 gittoken을 입력한다.

jenkins/*****

Add +

고급 v

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? 3.배포하고자 하는 브랜치를 선택한다.

*/BE

Add Branch

- 빌드 유발로 이동하여 webhook설정을 한다.

빌드 유발 체크 되었는대로 체크한다.

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: `http://f9d201.p.ssafy.io:9000/project/crit` ?

복사해서 가지고 있는다. 이는 gitlab webhook에 사용된다.

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 ?

Jenkins please retry a build

고급 ^ 클릭하여 추가 설정을 연다.

- ☒ Enable [ci-skip] ?
 - ☒ Ignore WIP Merge Requests ?
- Labels that launch a build if they are added (comma-separated) ?
-
- ☒ Set build description to build cause (eg. Merge request or Git Push) ?
 - ☐ Build on successful pipeline events

Pending build name for pipeline ?

- ☐ Cancel pending merge request builds on update ?

Allowed branches

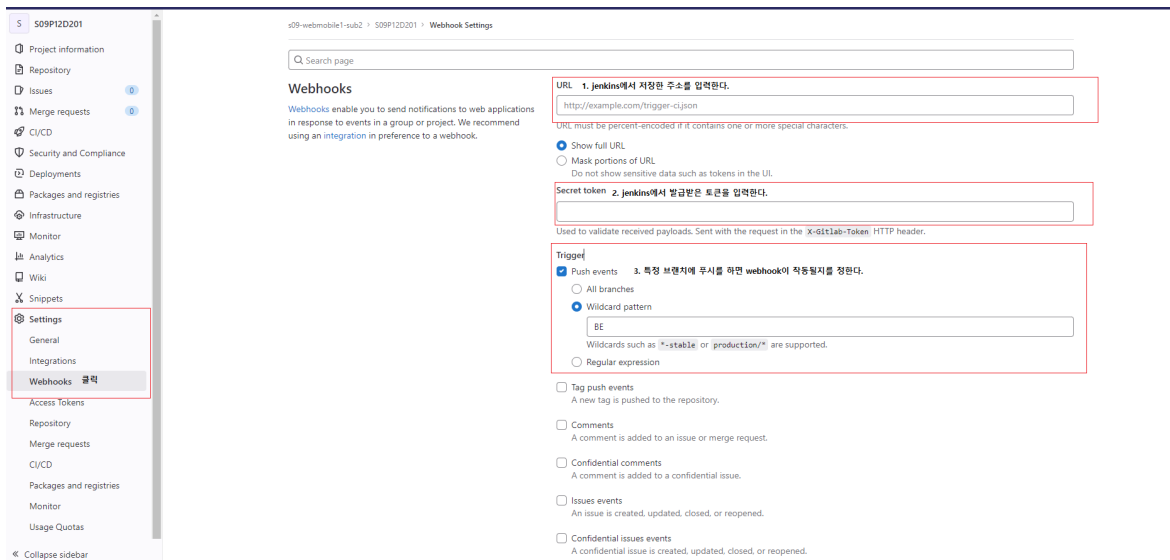
- ☒ Allow all branches to trigger this job ?
- ☐ Filter branches by name ?
- ☐ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token ? 이거만 확인하면 된다. Generate버튼을 클릭하여 새로운 토큰을 만들고 이를 아까 주소와 같이 저장한다.

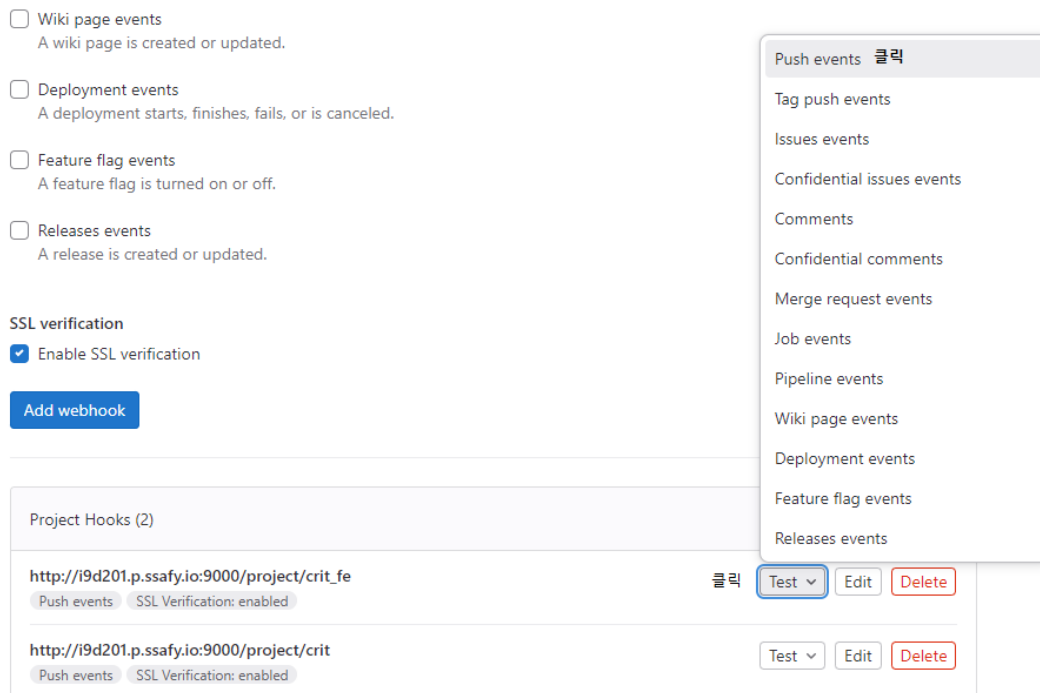
3dcf1c9f957f5f9630adcc30dfaaafe2

Generate

- gitlab에서 webhook을 등록하고 jenkins 와 연결한다.

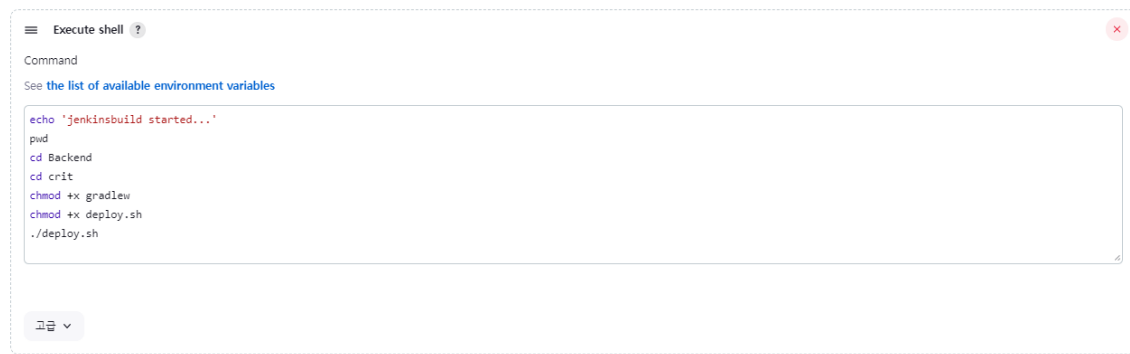


- 등록을 하면 밑에 나타나게 되는데 Test - push events를 클릭하면 테스트가 진행되고 위에 http 200이라는 말이 나오면 연결이 된 거다.



- build steps로 이동해 다음과 같이 입력한다.

Build Steps



```
echo 'jenkinsbuild started...'
pwd
cd Backend
cd crit
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh
```

- cd로 dockerfile과 deploy.sh가 있는 곳 까지 이동한다.
 - chmod +x 로 dockerfile과 deploy.sh에 읽기 권한을 준다.
 - ./deploy.sh로 실행한다.
 - gitlab에 push를 할 때 dockerfile과 deploy.sh를 같은 위치에 두어 ./deploy.sh라 하였지만 위치가 달라지면 맞는 위치로 변경해주면 된다.
- dockerfile의 내용

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
RUN chmod +x /app.jar # 이 부분을 추가하여 실행 권한을 부여합니다.
ENTRYPOINT ["sh", "-c", "java ${JAVA_OPTS} -jar /app.jar"]
```

- java 버전을 맞게 설정한다.
 - JAR_FILE이라는 변수를 설정하는데 이는 build가 되었을 때 나오는 jar파일을 지정해준다.
 - COPY로 복사하고 이에 대한 실행 권한을 부여한다.
 - jasypt를 사용하여 배포를 할 경우 해당 ENTRYPOINT를 동일하게 해야지 암호 키를 적용할 수 있다.
- deploy.sh의 내용

```
echo '실행 시작'

echo 'git pull'

echo 'jar 파일 삭제'
rm build/libs/*.jar

echo '빌드 시작'
./gradlew clean build

echo '도커파일 이미지 빌드'
docker build -t crit_be .

echo '컨테이너 중지'
docker stop crit_be

echo '컨테이너 삭제'
docker rm crit_be

echo '쓰레기 이미지 삭제'
```

```
docker images -f "dangling=true" -q | xargs docker rmi

echo '컨테이너 실행'
docker run -e JAVA_OPTS=-Djasypt.encryptor.password=asdkljfhawekjrawmnbvsjachgjhkermn12b34j2151872tdsfzsdbfkjaukjrahm -p 8080:8080
```

- echo '쓰레기 이미지 삭제'
 - 이 부분은 새로운 이미지가 만들어 질 때 전의 이미지가 삭제가 안되고 <none>라는 쓰레기 이미지로 저장에 계속 되어있어 용량에 영향을 끼치게 될 것같아 <none>라는 태그를 가진 모든 image를 삭제하는 명령어다
- echo '컨테이너 실행'
 - -e JAVA_OPTS이 부분을 사용하여 jasypt 비밀 키를 등록을 하여 컨테이너를 만들 때 안에 들어가게 만들었다.
 - —name을 하지 않으면 랜덤한 값의 이름으로 컨테이너가 만들어진다. 추후 유지보수를 위해 등록해 두는 것이 좋다.
 - -p는 ec2서버내에 열어둔 서버의 포트를 사용하여 컨테이너와 연결을 시켜준다.
- 마지막으로 ec2서버로 가서 sudo vim /etc/sudoers를 입력하여 해당 파일에 들어간다.

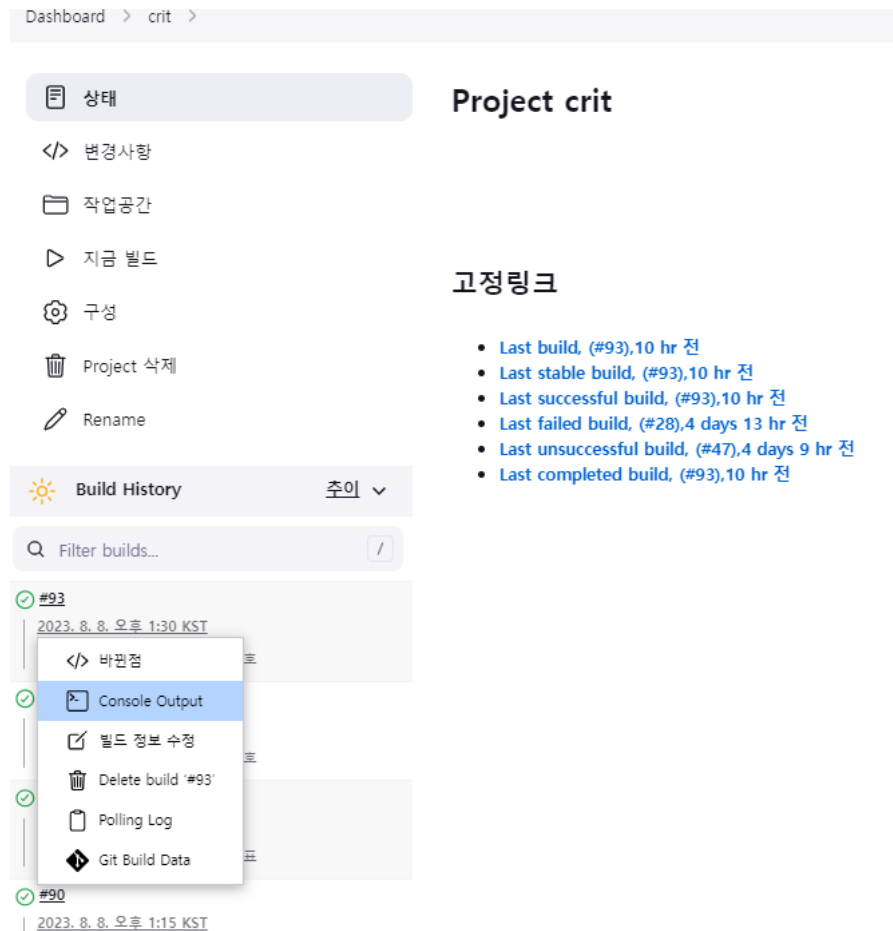
```
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
jenkins ALL=(ALL) NOPASSWD: ALL
# See sudoers(5) for more information on "#include" directives:

#include_dir /etc/sudoers.d

~
~
~
```

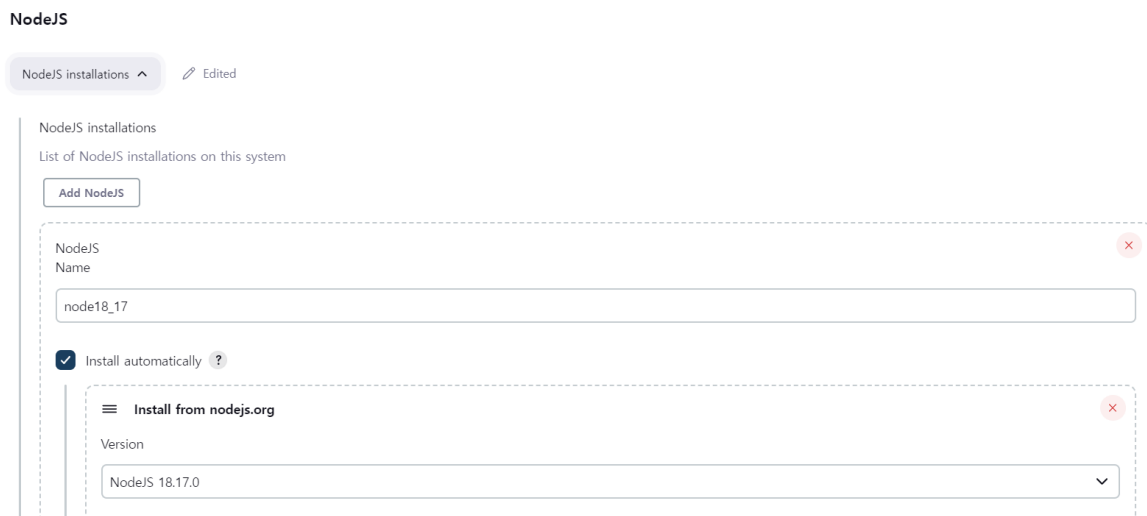
- 해당 부분에 jenkins ALL=(ALL) NOPASSWD: ALL 작성을 해주는데 이는 jenkins에 sudo권한을 부여하는 것이다.
- 이제 git의 연결해둔 브랜치에 push,merge를 하거나 혹은 jenkins에서 지금빌드를 클릭하면 새로운 docker 컨테이너가 생성되며 배포가 된다.
 - 혹시 새로운 컨테이너가 바로 꺼진다면 서버에서 docker logs <해당 컨테이너의 이름> 을 입력하면 log를 확인할 수 있고 이를 통해 어디에서 오류가 났는지 확인이 가능하다.
 - 혹시 jenkins가 제대로 돌아가지 않으면 jenkins 내의 console로 확인이 가능하다.



- 확인하고자 하는 빌드 번호를 클릭하거나 기다리면 나오는 오른쪽 화살표에 마우스를 대면 나오는 console Output을 클릭하면 해당 jenkins pipeline이 왜 실패했는지가 나오니 이를 보고 수정하면 된다.

5.3 Frontend


- 이제 jenkins에 들어가서 jenkins관리 → plugin에서 nodejs를 설치한다.
- 이후 tool에서 현재 ubuntu에 설치되어있는 node버전과 동일하게 추가한다.




- 이제 새로운 item을 만들어 주는데 백엔드 서버 배포 하는 것과 다르게 이번엔 pipeline으로 진행을 해보았다.

Enter an item name


» Required field




Freestyle project
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.



Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project
다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.



Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Multi-branch Pipeline

- 생성하고 구성으로 이동하여 백엔드와 동일하게 웹훅 설정을 해준다.

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://19d201.p.ssafy.io:9000/project/crit_fe ?

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 ?

- 이후 pipeline으로 이동하여 pipeline script를 클릭하여 다음과 같은 코드를 입력하고 저장하고 실행한다.

Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {  
2   agent any  
3  
4   tools {nodejs "node18_16_1"}  
5  
6   stages {  
7     stage('gitlab clone') {  
8       steps {  
9         git branch: 'FE', credentialsId: 'jenkins', url: 'https://lab.ssafy.com/s09-webmobile1-sub2/S09P12D201.git'  
10      }  
11    }  
12  
13    stage('build') {  
14      steps {  
15        dir('[FE]프론트/my-app') {  
16          nodejs(nodeJSInstallationName: 'node18_16_1') {  
17            sh 'npm install --force'
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

저장

Apply

```
pipeline {  
  agent any  
  
  tools {nodejs "node18_16_1"}  
  
  stages {  
    stage('gitlab clone') {  
      steps {  
        git branch: 'FE', credentialsId: 'jenkins', url: 'https://lab.ssafy.com/s09-webmobile1-sub2/S09P12D201.git'  
      }  
    }  
  
    stage('build') {  
      steps {  
        dir('[FE]프론트/my-app') {  
          nodejs(nodeJSInstallationName: 'node18_16_1') {  
            sh 'npm install --force'  
            sh 'npm run build'  
          }  
        }  
      }  
    }  
  
    stage('SSH-Server-EC2'){  
      steps {  
        echo 'SSH'  
  
        sshagent(credentials: ['ubuntu']) {  
          // Copy React build files  
          sh 'scp -r [FE]프론트/my-app/build/ ubuntu@172.26.5.19:/home/ubuntu/crit_fe'  
        }  
  
        mattermostSend (  
          color: "#D0E0E3",  
          icon: "https://jenkins.io/images/logos/jenkins/jenkins.png",  
          message: "프론트 배포가 성공했습니다."  
        )  
      }  
    }  
  }  
}
```

6. 외부서비스

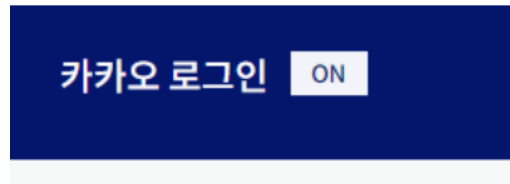
6.1 소셜 로그인 - kakao



✓ OAuth 기반 소셜 로그인 API 제공

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

- Kakao developer에서 로그인 후 애플리케이션 추가
- 카카오 로그인을 활성화



활성화 설정

상태

ON

카카오 로그인 API를 활용하면 사용자들이 번
상태가 OFF일 때도 카카오 로그인 설정 항목을
상태가 ON일 때만 실제 서비스에서 카카오 로

- 사이트 도메인, Redirect URI 등록

Web

사이트 도메인

- 카카오 로그인 사용 시 Redire

Redirect URI

Redirect URI

- 카카오 로그인에서 사용할 O

- 앱 키 메뉴에서 REST API 키를 가져와 git에 올리지 않을 yml 파일 작성
- 카카오 로그인 동의 항목 설정
- 외부로 노출 안 되도록 application-oauth.yml파일에 키와 관련 내용 작성

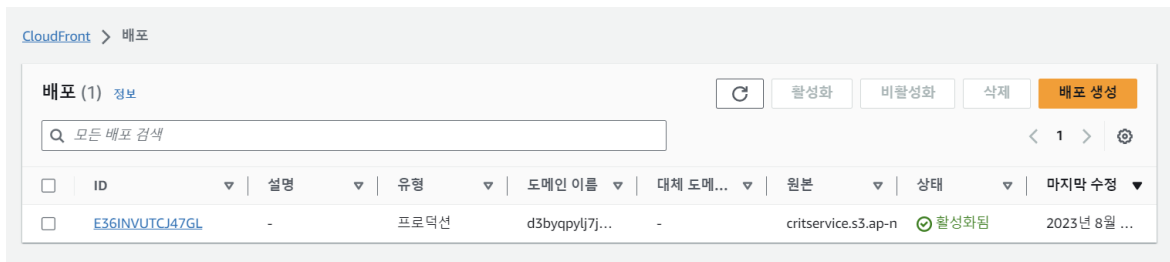
```
security:
  oauth2:
    client:
      registration:
        kakao:
          client-id: ENC(w/z6II7EsQwPdi8aX8pYUtcFXJK5kGULmt8kQrjPkuUuiH0t/CQ6JHL0TLZpCuLU)
          redirect-uri: ENC(w5I01CgaY0IBQI4GcVy3w54YgoemC214QXwRYaF7MmT7maEfp6jAw5mLs7dKGpj0cL++iteC7y4wL6JD7+BYA==)
          client-authentication-method: POST
          authorization-grant-type: authorization_code
          scope: profile_nickname, profile_image, account_email
          client-name: Kakao
```

- 카카오가 보내준 인가코드를 Front에서 Back으로 전달
- Back에서 카카오에게 인가코드를 전달하고 access-token 받아오기

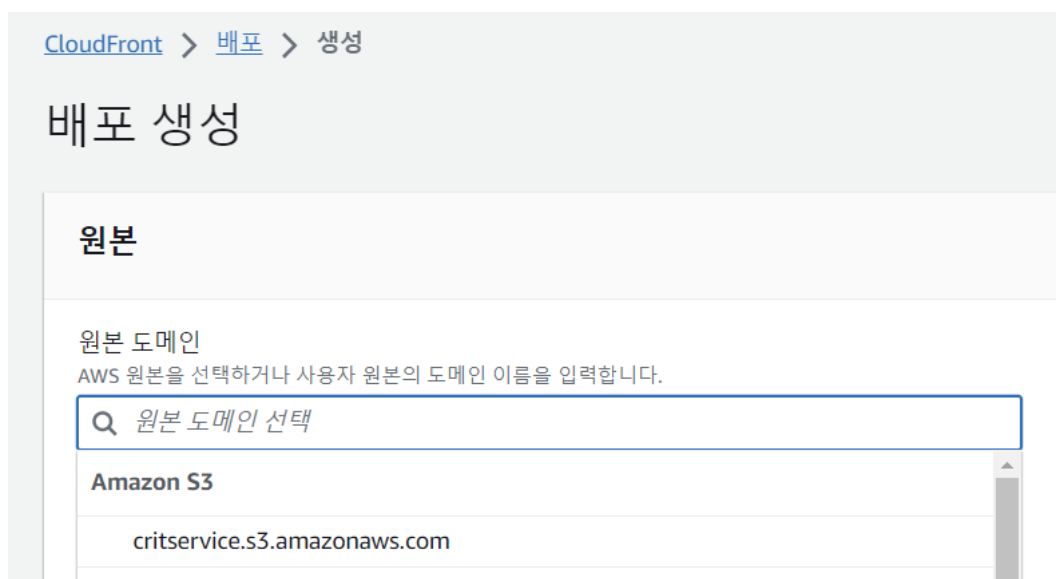
- access-token으로 카카오에 저장되어 있는 user 정보 를 받아오기

6.2 S3 & cloudfront

- aws에 로그인하여 CloudFront로 이동한다. 이후 배포 생성을 클릭



- 원본 도메인 선택을 클릭하여 나오는 s3주소를 클릭한다.



- 이후 나오는 원본 액세스를 원본 액세스 제어 설정(권장)으로 체크한다.

원본

원본 도메인

AWS 원본을 선택하거나 사용자 원본의 도메인 이름을 입력합니다.

critservice.s3.ap-northeast-2.amazonaws.com

원본 경로 - 선택 사항 | 정보

원본 요청의 원본 도메인 이름에 추가할 URL 경로를 입력합니다.

원본 경로 입력

이름

이 원본의 이름을 입력합니다.

critservice.s3.ap-northeast-2.amazonaws.com

원본 액세스 | 정보

☐ 공개

버킷은 공개 액세스를 허용해야 합니다.

☒ 원본 액세스 제어 설정(권장)

버킷은 CloudFront에 대한 액세스만 제한할 수 있습니다.

☐ Legacy access identities

CloudFront 원본 액세스 ID(OAI)를 사용하여 S3 버킷에 액세스합니다.

Origin access control

Select an existing origin access control (recommended) or create a new configuration.

Select an origin access control ▼

제어 설정 생성

- 이후 나오는 Origin access control에 있는 s3와 동일한 이름의 주소를 클릭한다. 혹시 에러가 나오거나 없는 경우 제어 설정 생성을 클릭 해준다.
 - 제어 설정 생성에 들어가서 특별히 건들 설정 없이 바로 생성 해주고 이를 사용해주다.

제어 설정 생성

이름

critservice.s3.ap-northeast-2.amazonaws.com

이름은 고유해야 합니다. 문자, 숫자 및 대부분의 특수 문자를 사용할 수 있습니다. 최대 64자까지 사용할 수 있습니다.

설명 - 선택 사항

설명 입력

설명은 최대 256자까지 입력할 수 있습니다.

서명 동작

☐ Do not sign requests
 ☒ Sign requests (recommended)

☐ 승인 헤더 재정의 안 함
수신 요청에 승인 헤더가 있는 경우 서명하지 마세요.

원본 유형

S3

원본 유형은 원본 도메인과 동일한 유형이어야 합니다.

취소

생성

- 이후 건물 설정 없이 바로 배포생성 버튼을 클릭해준다.
- 이후 생성한 cloudfront를 클릭하면 배포 도메인 이름이 나오는데 이를 복사하여 자신이 사용하고 있던 S3주소와 바꾸기만 하면 사용이 된다.

6.3 jasypt

- 의존성 추가

```
dependencies {
    implementation 'com.github.ulisesbocchio:jasypt-spring-boot-starter:3.0.4'
}
```

- 암호화 키 생성
 - <https://www.devglan.com/online-tools/jasypt-online-encryption-decryptio>

Jasypt Online Encryption

Enter plain text to Encrypt

Enter plain text to hash

Select Type of Encryption

One Way Encryption(Without Secret Text) ▾

Secret Key To Be Used While Encryption

Enter Secret Key

Encrypt

Encrypted String:

Result goes here

Jasypt Online Decryption

Enter the Jasypt Encrypted Text

Jasypt Encrypted Password

Select Action Type

Decrypt Password ▾

Secret Key Used during Encryption(Optional)

Enter Secret Key

Match/Decrypt

Result:

Result goes here

- Enter plain text to Encrypt : 암호화할 코드 입력
- Select Type of Encryption : 암호화할 방식 선택
- Secret Key to Be Used : 암호화에 사용할 시크릿 키 입력(원하는 아무 문자열로 입력)
- [Encrypt]를 누르면 암호화가 완료된 문자열이 아래 Encrypted String에 출력
- 복호화 config생성

```
package com.ssafy.crit.shorts.test;

import org.jasypt.encryption.StringEncryptor;
import org.jasypt.encryption.pbe.PooledPBEStringEncryptor;
import org.jasypt.encryption.pbe.config.SimpleStringPBEConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class JasyptConfig {

    @Value("${jasypt.encryptor.password}")
    private String password;

    @Bean("jasyptStringEncryptor")
    public StringEncryptor stringEncryptor() {
        PooledPBEStringEncryptor encryptor = new PooledPBEStringEncryptor();
        SimpleStringPBEConfig config = new SimpleStringPBEConfig();
        config.setPassword(password);
        config.setAlgorithm("PBKDF2WithHMACSHA512");
        config.setKeyObtentionIterations("1000");
        config.setPoolSize("1");
        config.setProviderName("SunJCE");
        config.setSaltGeneratorClassName("org.jasypt.salt.RandomSaltGenerator");
        config.setStringOutputType("base64");
        encryptor.setConfig(config);
        return encryptor;
    }
}
```

- 암호화한 내용을 ENC()안에 넣어서 수정

```
spring:
  servlet:
    multipart:
      max-file-size: 300MB
```

```

max-request-size: 300MB
datasource:
  url: ENC(FN1fGG/cz8fWQk7xLomjexe7tImW5KqXhw6NS1Qz9+lskpEV3JiYX6ilrS85fPGX8+v9gKwk5akrpAHkuR1+GyBxjsIwmGYikdjsGTvNDj87UFYb/YFP31H7T)
  username: ENC(cZTyh0FJelHqQilp2Y4lBwenULJBpF6U)
  password: ENC(UoG6QmblfdyyheAZHohflGtu0To7S+6s)
  driver-class-name: org.mariadb.jdbc.Driver
  hikari:
    maximum-pool-size:3
main:
  allow-circular-references: true

```

6.4 sonarqube

- 소나큐브 도커 이미지로 내려받고 컨테이너로 실행시키기

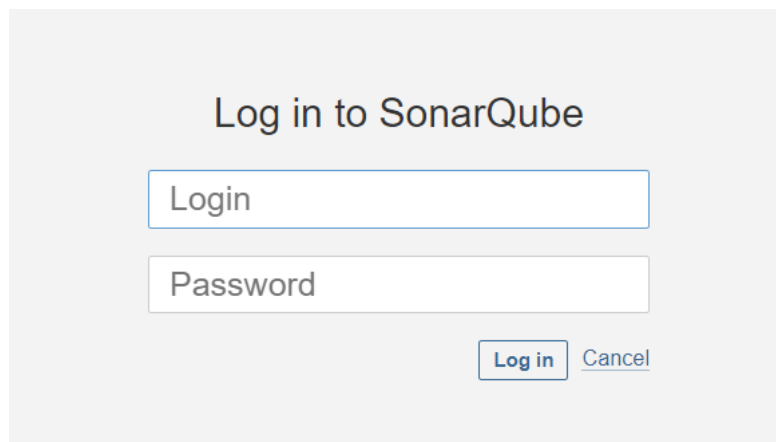
- `sudo docker pull sonarqube`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
crit_be	latest	57debfe61cd5	19 hours ago	807MB
sonarqube	latest	806094f6f03a	6 days ago	702MB
sonarqube	latest	48505167e4f0	2 weeks ago	676MB

- `sudo docker run -d --name sonarqube -p port:port sonarqube`

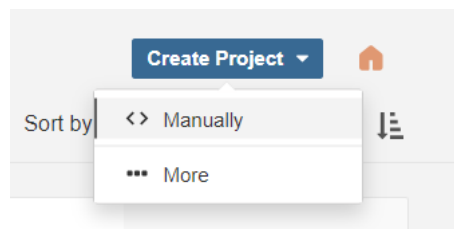
77f548d4928b	sonarqube	"/opt/sonarqube/docker/"	6 days ago	Up 6 days	0.0.0.0:8081->9080/tcp, :::8081->9080/tcp
--------------	-----------	--------------------------	------------	-----------	-------------------------------------------

- 연결한 도메인과 포트번호 입력하여 소나큐브 실행 ex) i9d201.p.ssafy.io:8081
 - 포트번호는 미리 연결해둘것
- 로그인(최초 로그인은 admin/admin으로 설정되어있음) → 추후 비밀번호 변경할 것



The image shows the SonarQube login interface. It has a title "Log in to SonarQube". Below the title are two input fields: "Login" and "Password". At the bottom right, there are two buttons: "Log in" and "Cancel".

- Manually 클릭하여 project 생성



- project display_name을 입력하면 project key와 동일하게 입력이 자동으로 됨

Create a project

All fields marked with * are required

Project display name *

 ✓

Up to 255 characters. Some scanners might override the value you provide.

Project key *

 ✓

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

The name of your project's default branch [Learn More](#)

[Next](#)

- Use the global setting 선택한 후 create project 클릭하여 생성

Set up project for Clean as You Code

New Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

What should be the baseline for new code for this project?

☒ Use the global setting

Previous version: Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version

Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases.

☐ Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code. Recommended for projects following continuous delivery.

☐ Reference branch

Choose a branch as the baseline for the new code. Recommended for projects using feature branches.

[Back](#) [Create project](#)

- 젠킨스로 돌아가서 jenkins관리 → plugins에서 sonarqube scanner 설치
- 이후 jenkins관리 → tool → sonarqube scanner에서 스캐너 버전을 선택
 - 이때 버전을 무조건 최신버전 5.0.1로 지정하지 말고 4.6.2 정도로 지정한다.
 - 5.0.1로 하면 java버전이 17로 지정이 되면서 에러가 생긴다.

- 이후 빌드하고자 하는 item으로 이동하여 구성을 클릭한다.
- 빌드 환경으로 이동하여 Use secret text(s) or file(s) 를 체크한다.
 - add 를 클릭하고 secret text를 선택한다. 이후 Credentials 부분에서 등록된 sonarqube를 선택한다.

빌드 환경

- ☐ Delete workspace before build starts
- ☒ Use secret text(s) or file(s) ?

Bindings

- Build steps로 넘어가 add build step을 선택하고 execute sonarqube scanner를 클릭한다.
- 이후 analysis properties부분에

```
sonar.projectKey=crit_Backend

sonar.exclusions=**/*.java

sonar.coverage.exclusions=**/*.java
```

해당 내용을 추가하고 빌드를 시작하면 sonarqube가 실행되며 bug를 찾아준다.

Execute SonarQube Scanner

Task to run ?

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

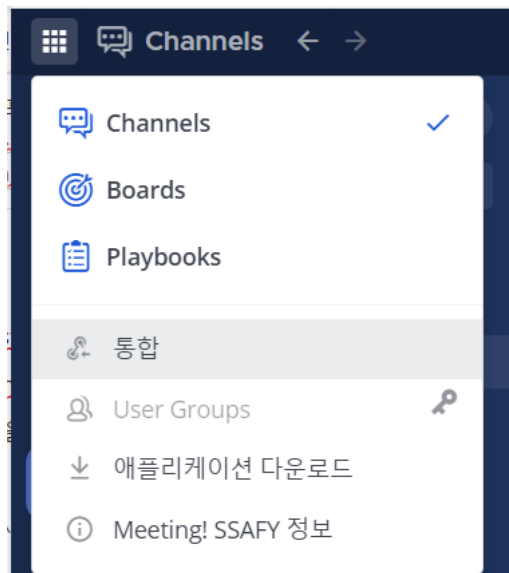
Analysis properties ?

sonar.projectKey=crit_Backend
sonar.exclusions=**/*.java
sonar.coverage.exclusions=**/*.java

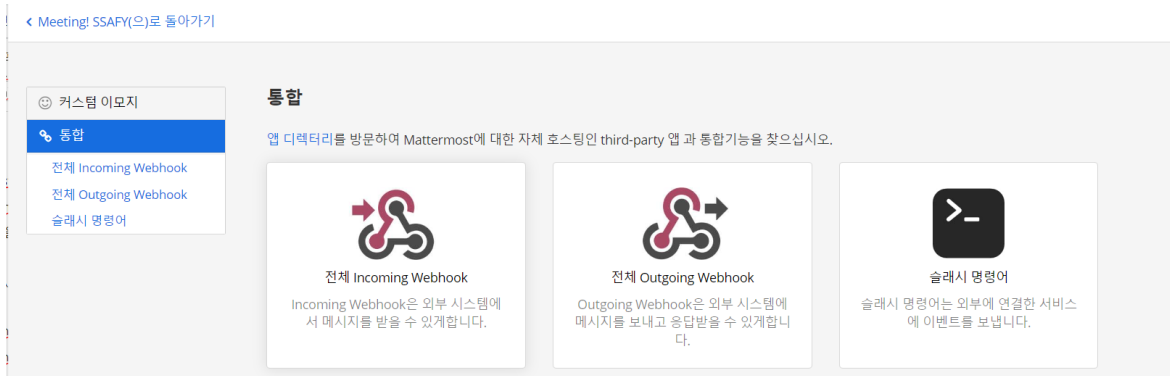
💡 java위치를 찾지못한다는 오류가 발생할 수 있는데 이는 jenkins의 jdk위치를 설정을 해주면 된다.
jenkins관리 → tool → jdk로 이동해서 이름은 원하는 이름으로 입력하고 JAVA_HOME에는 /opt/java/openjdk 로 입력을한
다. 이 주소는 ubuntu서버 내의 자바의 위치이다.
이렇게 설정하고 jenkins로 프로젝트를 빌드하면 sonarqube가 실행이 된다.

6.5 mattermost

- mattermost에서 좌측 상단에 있는 버튼을 눌러 통합으로 이동합니다.



- 전체 Incoming Webhook → Incoming Webhook 추가하기 클릭



- 제목, 설명을 적당히 입력 후 알림을 받아 볼 채널을 선택합니다.
 - 채널이 없으면 직접 하나를 만들면 된다.

전체 Incoming Webhook > 추가

제목

웹훅 설정 페이지에 대해 최대 64자의 제목을 지정합니다.

설명

웹훅에 대한 설명을 입력하세요.

채널

--- 채널을 선택하세요 ---

웹훅 페이로드를 수신할 기본 채널(공개 혹은 비공개)입니다. 비공개 채널로 웹훅을 설정할 때에는 그 채널에 속해 있어야 합니다.

이 채널로 고정

☐

설정되면, 들어오는 웹훅은 선택된 채널에만 게시할 수 있습니다.

취소

저장

- 목록에서 새로운 webhook이 생성된 것을 확인할 수 있는데 여기서 나온 URL을 복사한다.

Incoming Webhook 추가하기

Incoming webhook을 사용하여 외부 도구를 Mattermost에 연결 [Build Your Own](#) 또는 [앱 디렉터리](#)에 방문하여 자체 호스팅 third-party 앱과 통합기능을 찾습니다.

koo3435(이)가 2023년 8월 9일 수요일에 생성

- Apply

- 28

빌드 후 조치

Filter

Aggregate downstream test results
 Archive the artifacts
 Build other projects
 Publish JUnit test result report
 Record fingerprints of files to track usage
 Stop Docker Containers
 Git Publisher
 Accept GitLab merge request on success
 Add note with build status on GitLab merge requests
 Add vote for build status on GitLab merge requests
 E-mail Notification
 Editable Email Notification
 Mattermost Notifications
 Publish build status to GitLab
 Set GitHub commit status (universal)
 Set build status on GitHub commit [deprecated]
 Delete workspace when build is done

빌드 후 조치 추가 ▲

- 받고자 하는 알림 선택 후 저장

빌드 후 조치

Mattermost Notifications

☐ Notify Build Start
☐ Notify Aborted
☐ Notify Failure
☐ Notify Not Built
☒ Notify Success
☐ Notify Unstable
☐ Notify Back To Normal

고급 ▼