

**NONING 포팅 메뉴얼**

A202 다섯논쟁이

**목차**

[I. 개요 2](#_Toc111764546)

[1. 프로젝트 개요 2](#_Toc111764547)

[2. 프로젝트 사용 도구 2](#_Toc111764548)

[3. 개발 환경 2](#_Toc111764549)

[4. 외부 서비스 3](#_Toc111764550)

[II. 빌드 7](#_Toc111764551)

[1. 환경변수 형태 7](#_Toc111764552)

[2. 빌드하기 9](#_Toc111764553)

[3. 배포하기 9](#_Toc111764554)

# 개요

## 프로젝트 개요

‘애인이 이성친구의 깻잎 떼어주는거 가능?’ 깻잎 논쟁에 대한 당신의 생각은 어떤가요? 다른 사람들의 생각은 궁금하지 않나요? 이걸 고를까 저걸 고를까? 고민되는 다양한 논쟁들에 2022년의 핫한 트렌드 ShortFun을 결합한 논잉을 만나보세요.

논잉은 쉽고 빠르게 즐기는 다양한 논쟁거리들과 다른 사람들의 의견을 볼 수 있는 분석페이지, 같은 논쟁에 대해 얘기를 나눌 수 있는 실시간 음성 채팅 등 다양한 기능을 가지고 있는 ‘본격 편가르기 논쟁 SNS’입니다.

논잉과 함께 재미있는 논쟁을 시작해보세요. 논쟁을 멈추지 마! 논잉.

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

UCC : 모바비

CI/CD : Jenkins, Docker

## 개발 환경

Frontend – React Native

Visual Studio Code IDE 1.69.0

Android Studio Chipmunk : 2021.2.1 Patch 2

react-native : ^0.69.2

AVD

Android SDK : 30

Android : 11.0 Google APIs

Model : Pixel 5 API 30

Backend – Spring

IntelliJ IDEA : 2021.3.1

IntelliJ Runtime: 11.0.13+7-b1751.21 amd64

VM : OpenJDK 64-Bit Server VM by JetBrains s.r.o

JDK : 11.0.15

Backend – Node

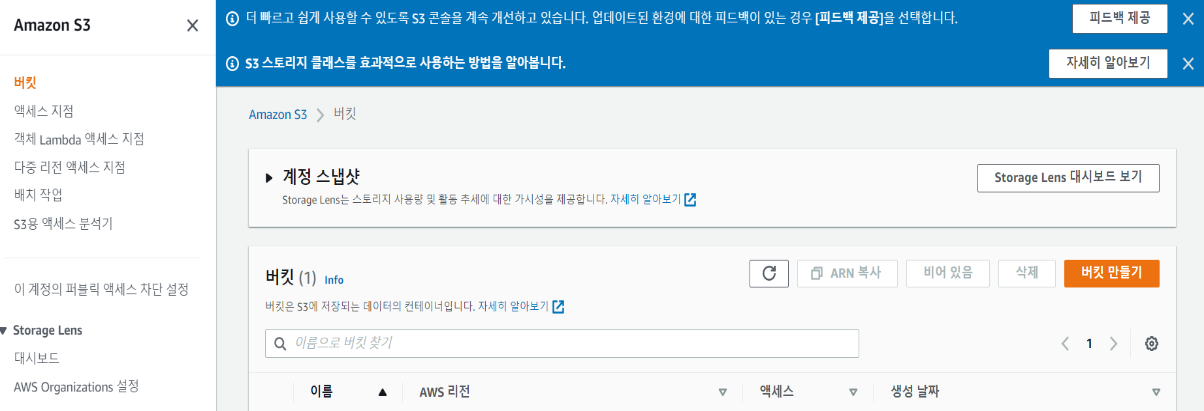
Node.js : 14.20.0

Python : 3.9

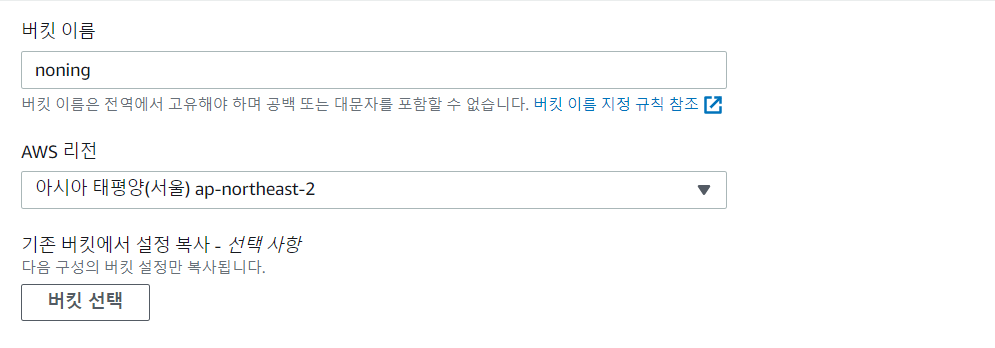
## 외부 서비스

Amazon Web Service S3 (Simple Storage Service)

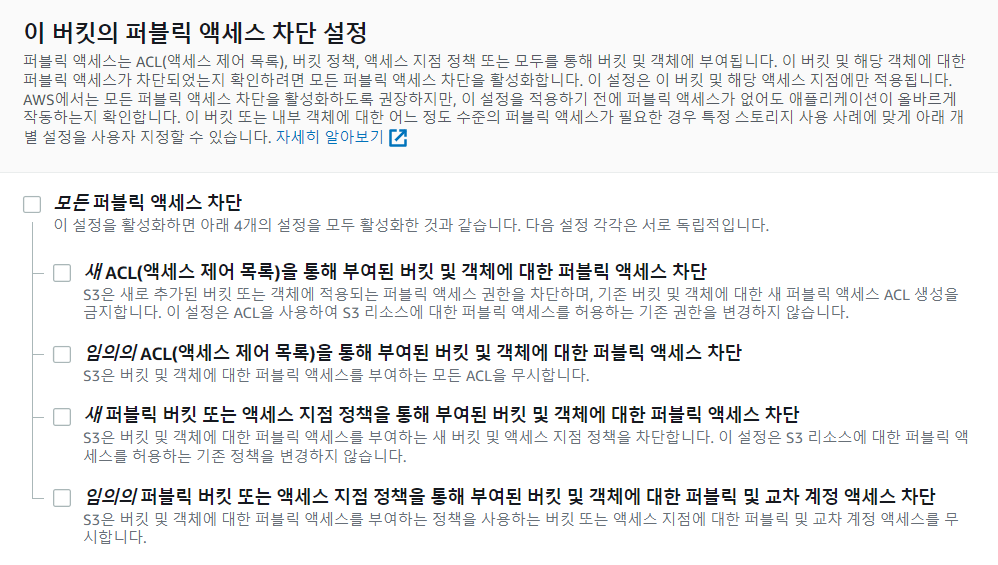
1. AWS 가입하기
2. S3 버킷 생성
   1. 버킷 만들기 선택



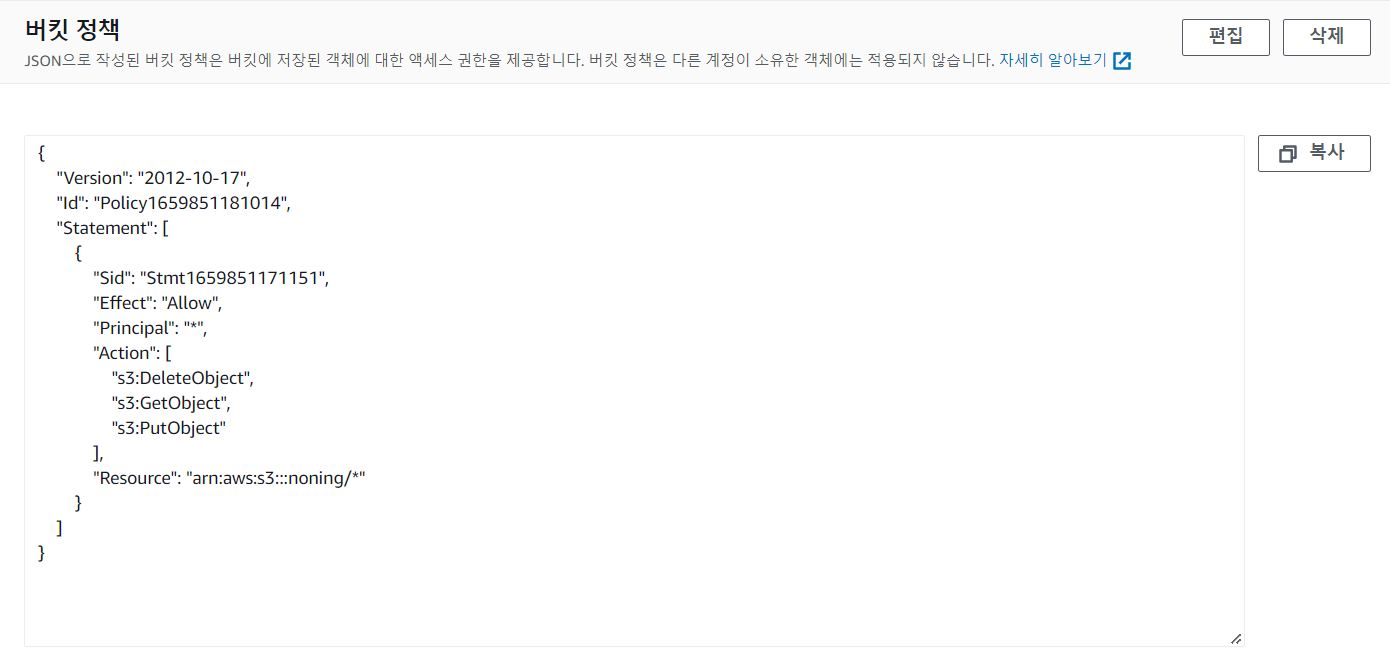
* 1. 버킷 이름 작성 및 AWS리전 서울 선택



* 1. 퍼블렉 엑세스 차단 풀어주기



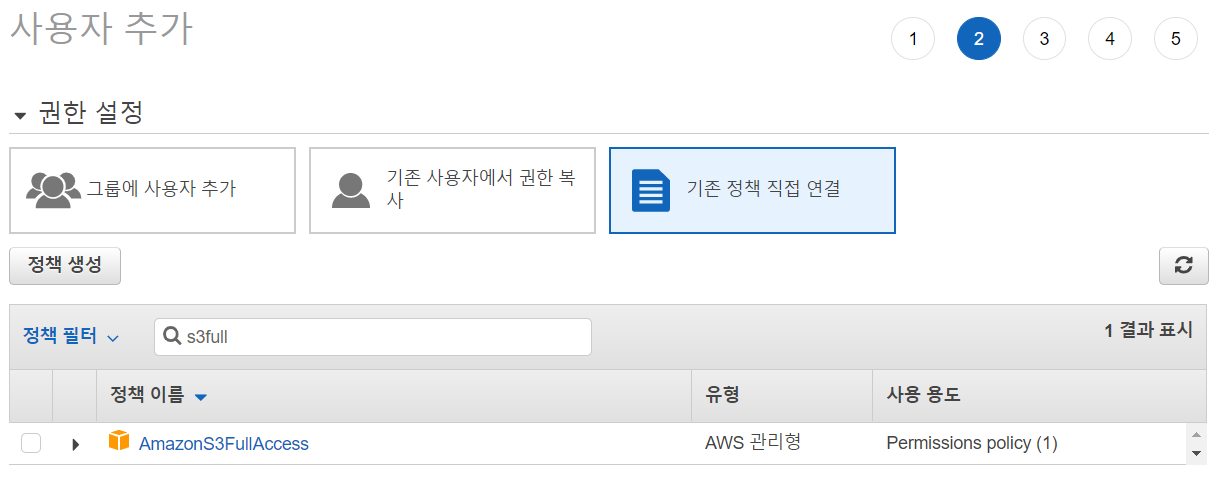
* 1. 버킷 만들기
  2. 버킷 – 권한에서 버킷 정책 설정



1. IAM 추가



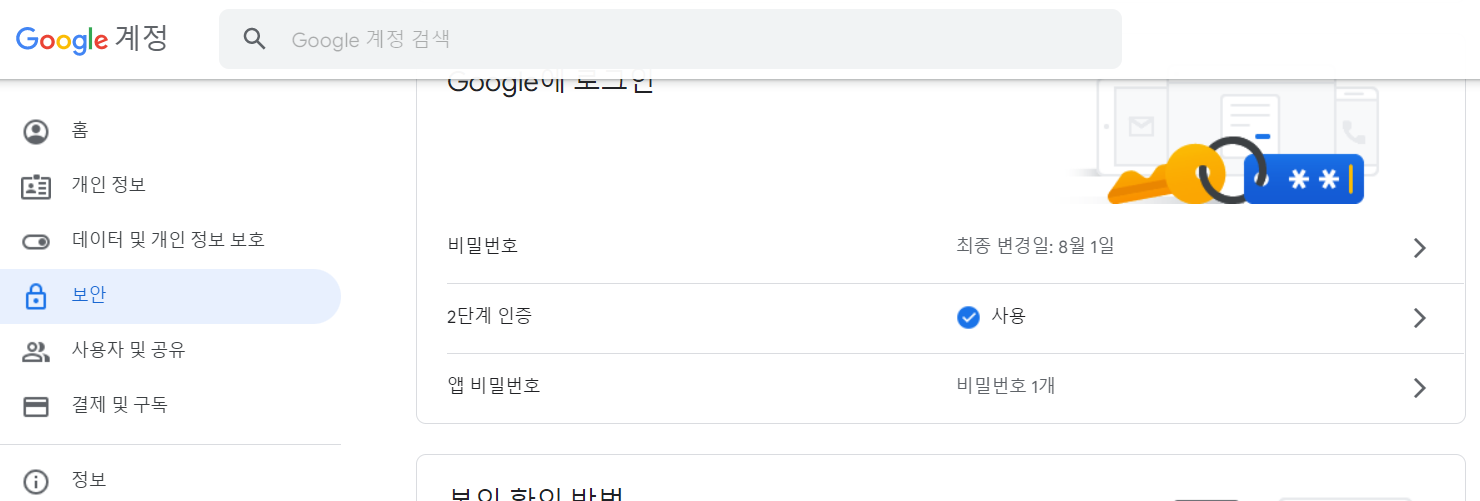
* 1. IAM 사용자 추가 – 엑세스 키 체크



* 1. S3FullAccess 정책 연결
  2. 생성된 사용자 csv파일 받은 후 Access Key, Secret key 확인

Google Gmail SMTP

1. Google 메일 앱 비밀번호 발급
   1. Google 계정 설정 - 2단계 인증 설정



* 1. Google 계정 설정 – 보안 – 앱 비밀번호



* 1. 메일 , Windows 컴퓨터 비밀번호 생성
  2. Windows 컴퓨터용 앱 비밀번호 확인

1. POP/IMAP 설정
   1. Gmail – 모든 설정



* 1. 전달 및 POP/IMAP 설정

# 빌드

## 환경변수 형태

application.properties

# Maria DB

spring.datasource.driverClassName=org.mariadb.jdbc.Driver

spring.datasource.url : DB 주소

spring.datasource.username : 유저 이름

spring.datasource.password : 유저 비밀번호

# JPA

spring.jpa.hibernate.ddl-auto : update (변경되는 값만 적용 \*운영 DB에 적용 X)

spring.jpa.show-sql : true (쿼리 로그 확인)

# JWT

JWT.SECRET : 시크릿 키

# Redis

spring.redis.host : 호스트 주소

spring.redis.password : 비밀번호

spring.redis.port : 포트

#SMTP

spring.mail.host : 주소 (구글 SMTP : smtp.gmail.com)

spring.mail.port : 포트 (구글 SMTP : 587)

spring.mail.username : 구글 ID

spring.mail.password : 메일 접속 비밀번호

spring.mail.properties.mail.smtp.auth : auth 명령 사용여부 ( true )

spring.mail.properties.mail.smtp.starttls.enable : TLS-protection 사용여부 ( true )

#WEBHOOK

notification.mattermost.enabled : 사용여부 ( true )

notification.mattermost.webhook-url : Webhook 주소

notification.mattermost.pretext : attachment의 상단에 나오는 텍스트

notification.mattermost.author-name : attachment의 상단에 나오는 이름

#S3 Bucket

cloud.aws.credentials.accessKey : 버켓 접근키

cloud.aws.credentials.secretKey : 버켓 비밀키

cloud.aws.stack.auto : 스택이름 자동 감지여부 ( false )

cloud.aws.s3.bucket : 버켓 이름

cloud.aws.region.static : 버켓 지역 (ap-northeast-2)

#multipart 사이즈 설정

spring.servlet.multipart.max-file-size=20MB

spring.servlet.multipart.max-request-size=20MB

## 빌드하기

Frontend

npm i

npx react-native run-android

Backend – Spring ( port : 8888 )

Gradle 빌드

Bootjar

Bootrun

Backend – Node (port 3000, 8400-8880)

npm i

npm start

## 배포하기

**Nginx**

Nginx 설정

|  |
| --- |
| server {  listen 80;  listen [::]:80;  server\_name i7a202.p.ssafy.io;  rewrite ^(.\*)https://i7a202.p.ssafy.io$1 permanent;  }  server {  listen 443 ssl;  listen [::]:443;    server\_name i7a202.p.ssafy.io;  # SSL configuration  ssl\_certificate /etc/letsencrypt/live/i7a202.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/i7a202.p.ssafy.io/privkey.pem;    location /api {  proxy\_pass http://i7a202.p.ssafy.io:8888;  }  } |

Nginx 재실행

|  |
| --- |
| sudo service nginx restart |

**Docker 설치**

필요 패키지 설치

|  |
| --- |
| sudo apt install apt-transport-https ca-certificates curl software-properties-common |

레포지토리 추가

|  |
| --- |
| curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable" |

Docker 설치

|  |
| --- |
| sudo apt install docker-ce |

Docker compose 설치

|  |
| --- |
| sudo curl -L https://github.com/docker/compose/releases/download/v2.1.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose |

권한 설정

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

**Redis 설치**

Redis 이미지 받아오기

|  |
| --- |
| docker image pull redis |

Redis 8379포트로 실행 및 비밀번호 설정

|  |
| --- |
| docker run -d –name noning\_redis -p -8379:6379 redis –requirepass “비밀번호” |

**Jenkins 설치**

Docker-compose.yml 파일 작성

|  |
| --- |
| version: "3"  services:  jenkins:  container\_name: jenkins-compose  image: jenkins/jenkins:lts-jdk11  build:  context: jenkins-dockerfile  dockerfile: Dockerfile  user: root  ports:  - "9090:8080"  volumes:  - /home/noning/compose/jenkins:/var/jenkins\_home  - /var/run/docker.sock:/var/run/docker.sock  - /usr/bin/docker:/usr/bin/docker  - /var/www/i7a202.p.ssafy.io/noning:/var/www/i7a202.p.ssafy.io/noning  privileged: true |

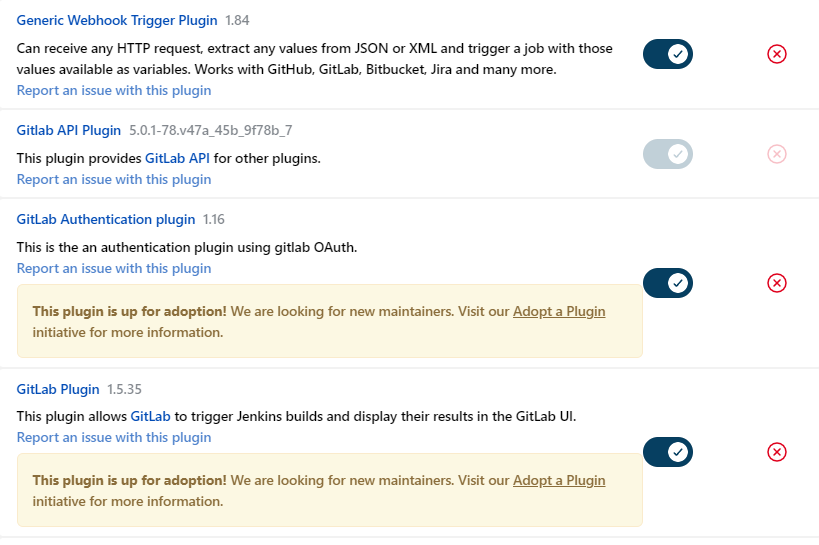
Docker-compose 실행

|  |
| --- |
| docker-compose up -d |

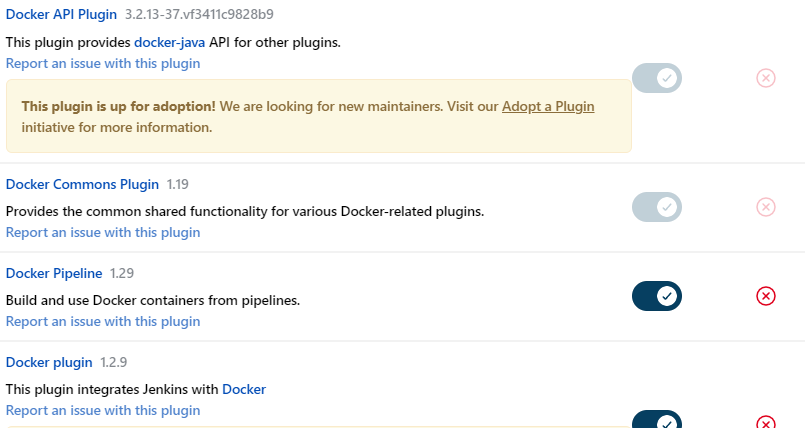
모든 Docker container 확인

|  |
| --- |
| docker ps -a |

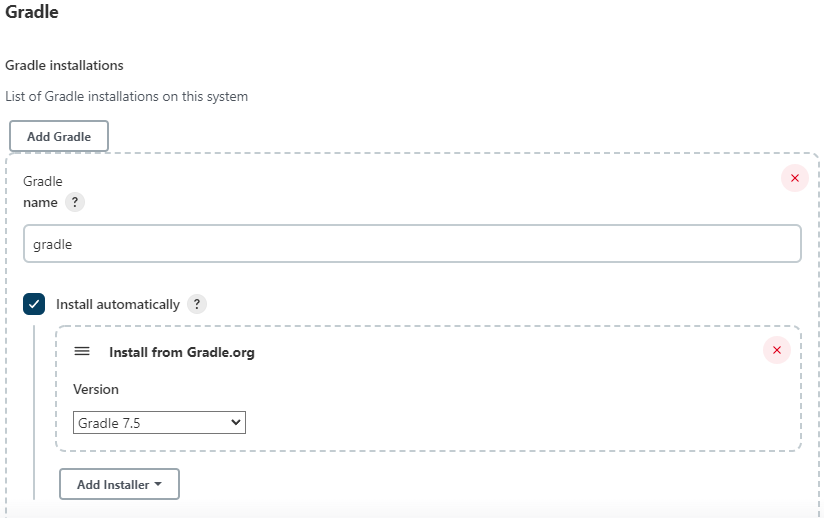
Gitlab Plugin 설치



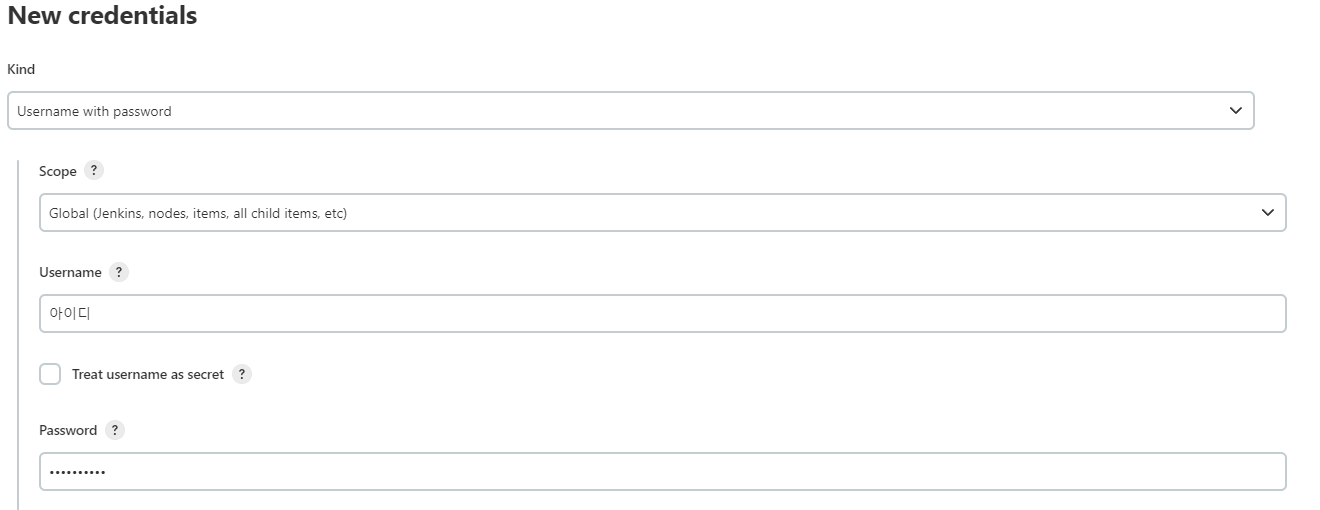
Docker Plugin 설치



Graddle 설정

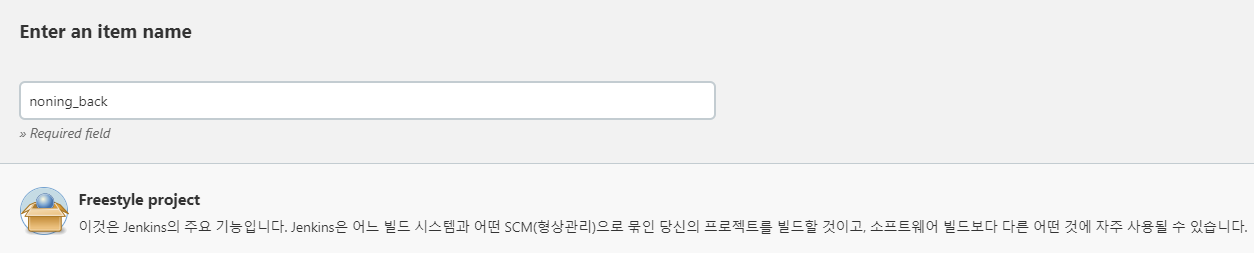


Git Account 설정

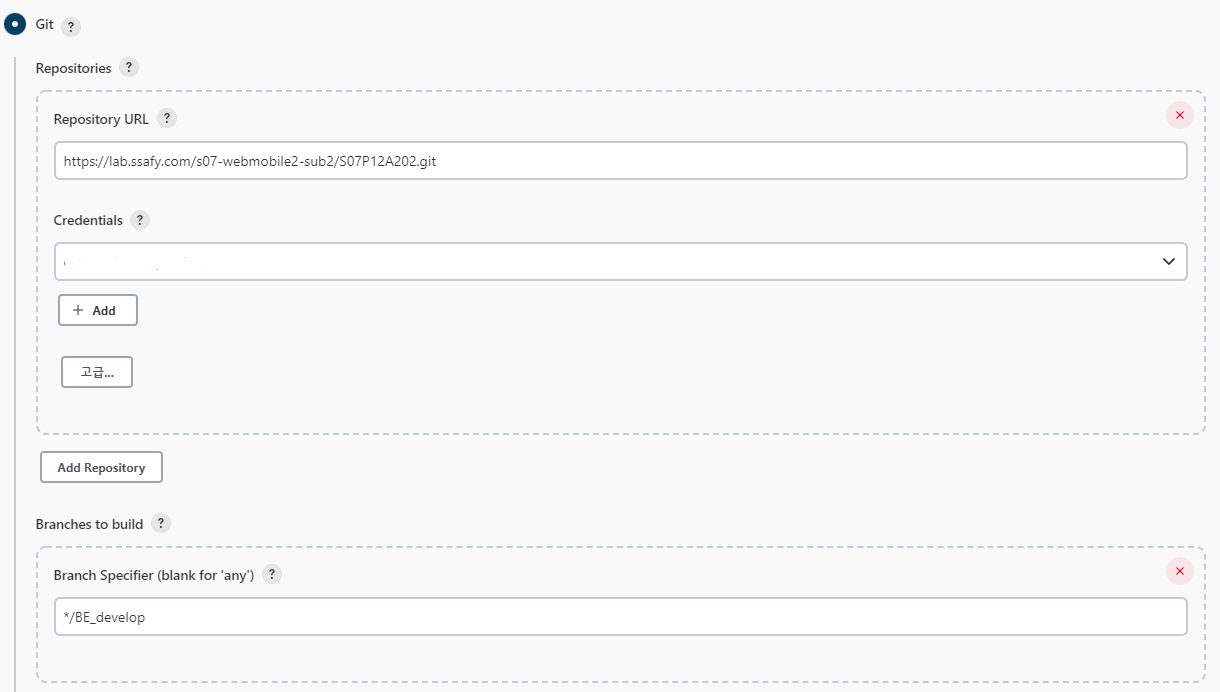


**Backend - Spring 빌드 (w. Jenkins)**

아이템 생성



Git 설정



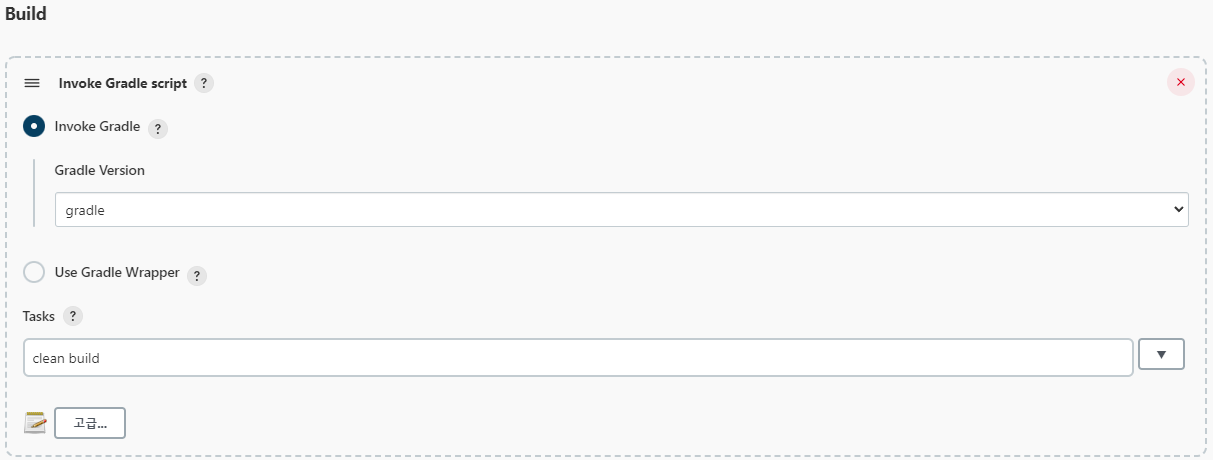
빌드 유발 설정





Secret Token 복사하기

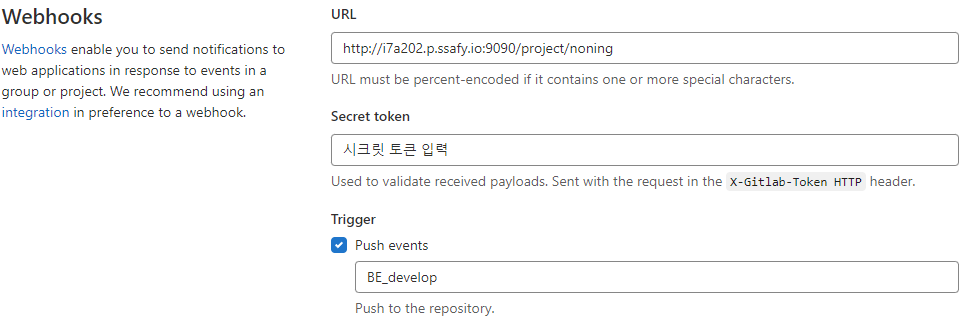
Graddle로 빌드



|  |
| --- |
| cd ./backend  docker build -t noning:latest .  docker stop noning\_back  docker rm noning\_back  docker rmi $(docker images -f "dangling=true" -q)  docker run --name noning\_back -d -p 8888:8888 -p 8400-8880:8400-8880 noning:latest |

Excute Shell 설정

Git WebHook설정



**Backend – Node 빌드 (w. Jenkins)**

아이템 생성, Git 설정, 빌드 유발, Git WebHook은 Spring과 동일 (branch만 변경)

Excute Shell 설정

|  |
| --- |
| cd ./chatserver  docker build -t noningchat:latest .  docker stop noning\_chat  docker rm noning\_chat  if [ $(docker images -f dangling=true -q) ]  then  docker rmi $(docker images -f "dangling=true" -q)  else  echo ""  fi  docker run --name noning\_chat -d -p 3000:3000 -v /var/www/i7a202.p.ssafy.io/noning:/var/www/i7a202.p.ssafy.io/noning noningchat:latest |

**Frontend apk 빌드**

|  |
| --- |
| cd android  ./gradlew assemblerelease |

APK release 파일 생성

APK 파일 위치

|  |
| --- |
| /android/app/build/outputs/apk/release |