## 가로선



설치 및 구성 메뉴얼

2023년 8월 19일

**─**

김정훈

# 프로젝트 Vagrantfile 구성

<https://github.com/KimJeongHoon190/First_Project/blob/main/project1/Vagrantfile>



# 

# 

# 

# 파이프라인 작업 리포지토리

<https://github.com/KimJeongHoon190/flask-cicd>

**Jenkinsfile 구성**

<https://github.com/KimJeongHoon190/flask-cicd/blob/main/.jenkins/Jenkinsfile>

# Docker-registry 설치

## Docker 설치

Docker-registry설치 전 준비물 : Ubuntu 20.04 환경에서 Docker 설치 완료 >>

**Docker 기본 패키지 설치**

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

**Docker에서 제공하는 패키지 저장소 등록**

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o

/etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

**패키지 저장소 경로 GPG 키와 함께 등록**

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

**등록한 패키지 저장소 적용**

sudo apt-get update

**Docker 설치**

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

**설치 후 확인**

docker -v

**vagrant가 docker 명령어를 사용할 수 있게 권한 수정**

sudo usermod -a -G docker vagrant

newgrp docker

groups

**Docker Hub 사이트에 로그인을 한 뒤 Docker Token 발급받기**

**Docker Token을 가져와서 로컬 파일로 저장하기**

생성된 Docker Token을 /vagrant/env/docker\_token 파일에 저장한다.

**Docker Token으로 로그인 성공하기**

docker login -u hedgehoon --password-stdin < /vagrant/env/docker\_token

2. Docker-registry 서버 기능 구현

**Docker-registry 서버에서 >>**

docker pull registry:latest

#docker-registy 서버를 구성하기 위한 필수적인 이미지

docker run -d -p 5000:5000 --restart=always --name registry registry:latest

# 다운받은 registry:latest 이미지로 registry라는 이름의 컨테이너를 실행

docker logs registry

# 해당 컨테이너가 제대로 작동하는지 로그 기록 확인

**Jenkins 서버(도커 레지스트리 클라이언트) 에서 >>**

sudo vi /etc/hosts

* 추가

<Docker\_registry\_server\_IP> docker-registry

ping docker-registry

# 통신 확인

sudo vi /etc/docker/daemon.json

{

"insecure-registries":["docker-registry:5000"]

}

sudo systemctl restart docker

# 데몬 파일 수정했으니 도커 재시작

docker pull jenkins/Jenkins:2.387.2-lts

# 도커 허브에서 설치에 필요한 이미지 원본 다운받기

docker tag jenkins/jenkins:2.387.2-lts

docker-registry:5000/my-jenkins-image:2.387.2-lts

#다운받은 원본 이미지 리태그

docker push docker-registry:5000/my-jenkins-image:2.387.2-lts

# docker-registry 서버로 리태그한 이미지 Push

**Docker-registry 서버에서 >>**

docker pull docker-registry:5000/my-jenkins-image:2.387.2-lts

# 방금 전에 Push한 이미지 받기

vagrant@docker-registry:~$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

registry latest 0030ba3d620c 2 weeks ago 24.1MB

docker-registry:5000/my-jenkins-image 2.387.2-lts 056e9a39e8be 4 months ago 471MB

# Jenkins 설치

## 1. Jenkins 서버 설치

Jenkins 설치 전 준비물 : Ubuntu 20.04 환경에서 Docker 설치 완료 >>

**Docker 기본 패키지 설치**

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

**Docker에서 제공하는 패키지 저장소 등록**

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o

/etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

**패키지 저장소 경로 GPG 키와 함께 등록**

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

**등록한 패키지 저장소 적용**

sudo apt-get update

**Docker 설치**

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

**설치 후 확인**

docker -v

**vagrant가 docker 명령어를 사용할 수 있게 권한 수정**

sudo usermod -a -G docker vagrant

newgrp docker

groups

**Docker Hub 사이트에 로그인을 한 뒤 Docker Token 발급받기**

**Docker Token을 가져와서 로컬 파일로 저장하기**

생성된 Docker Token을 /vagrant/env/docker\_token 파일에 저장한다.

**Docker Token으로 로그인 성공하기**

docker login -u hedgehoon --password-stdin < /vagrant/env/docker\_token

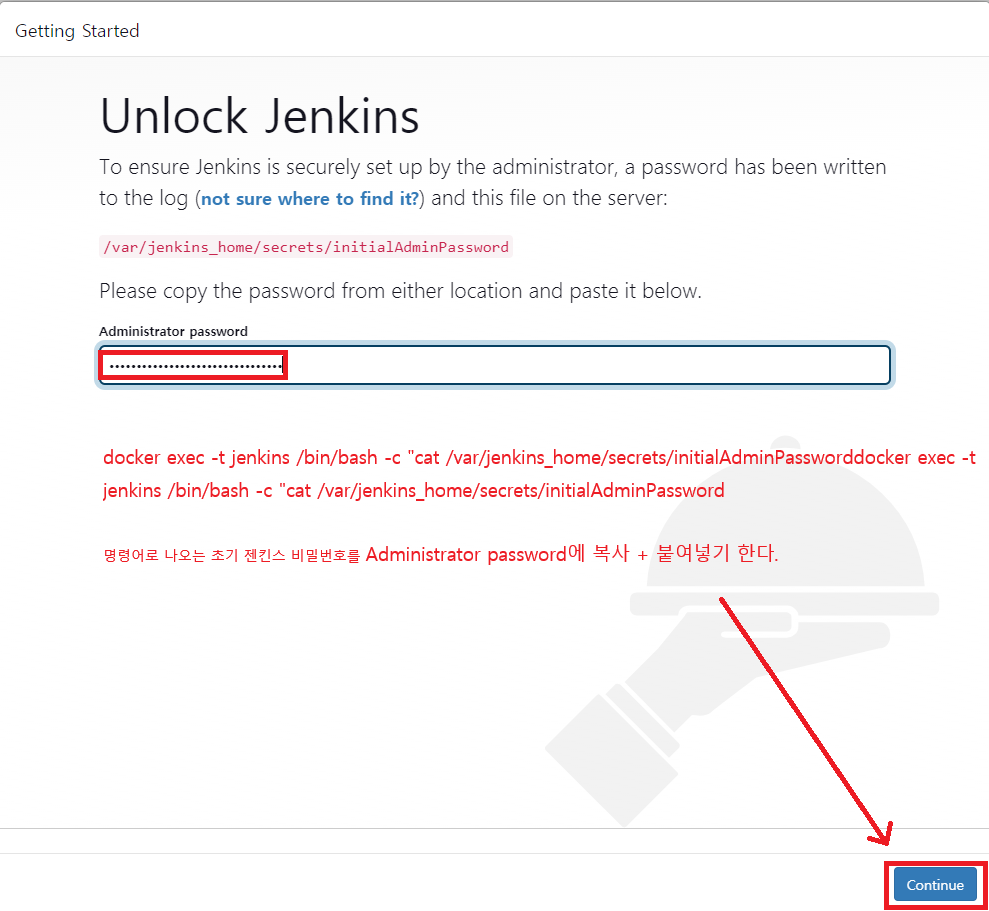
docker volume create jenkins-volume

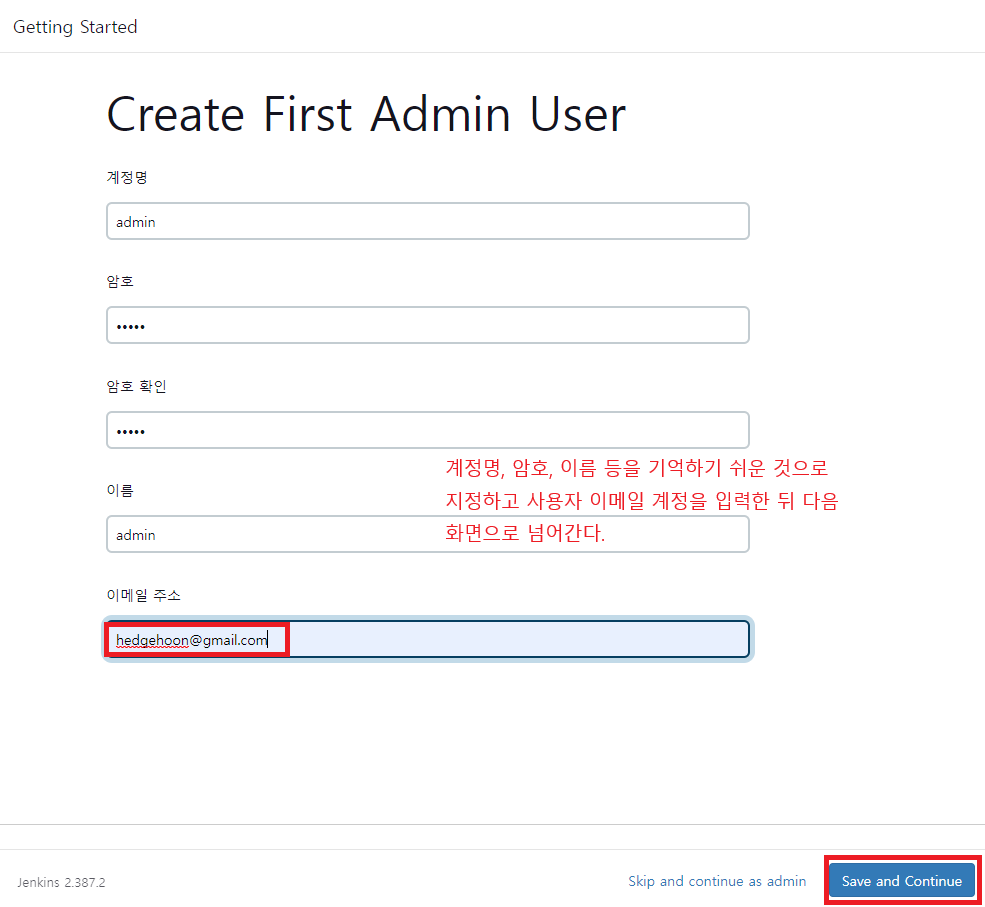
docker volume ls

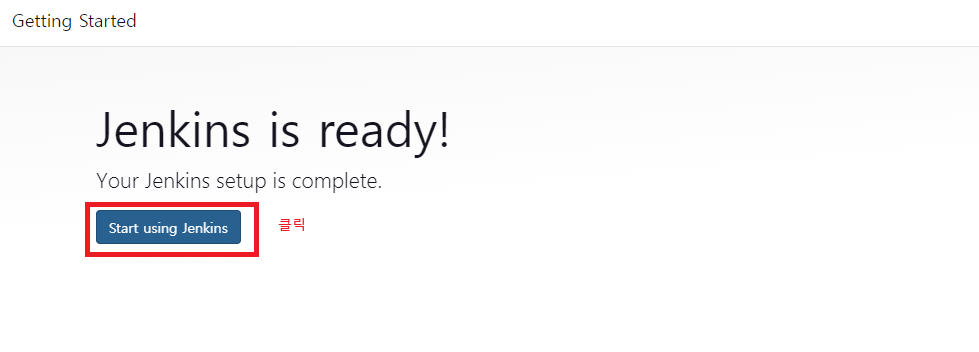
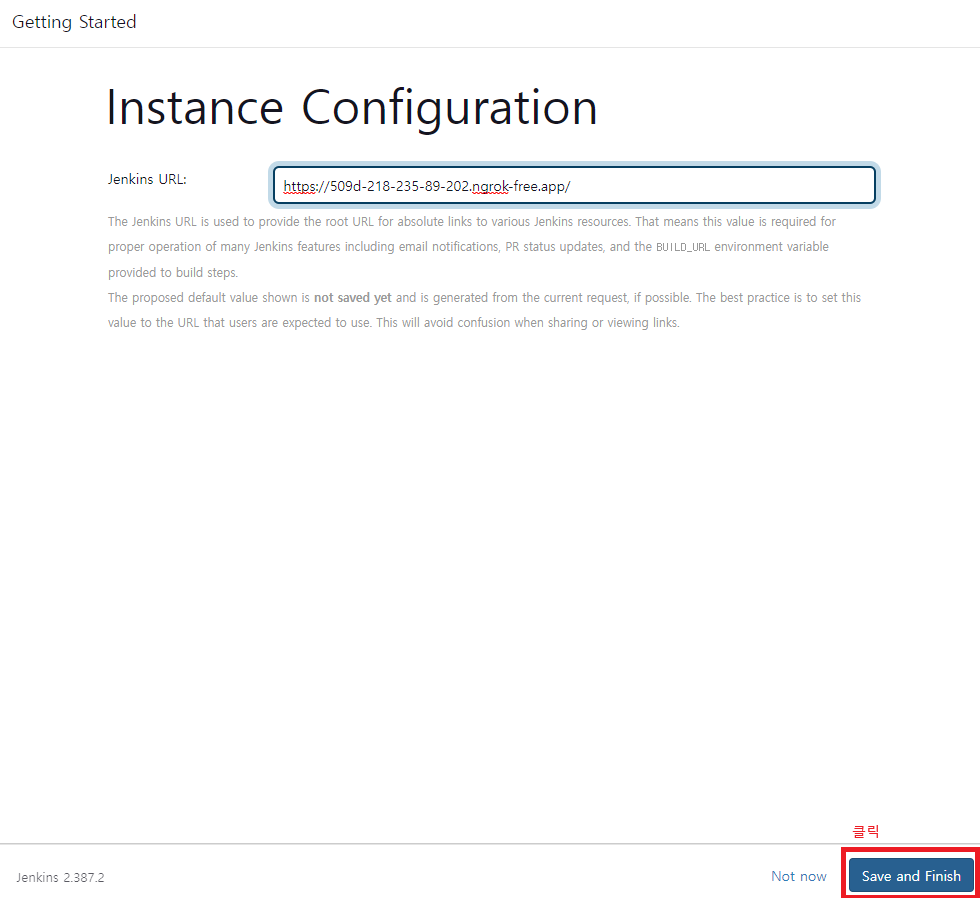
docker run -it -d -p 8080:8080 --restart=always --name jenkins -v jenkins-volume:/var/jenkins\_home/ -v /var/run/docker.sock:/var/run/docker.sock -v $(which docker):/usr/bin/docker --group-add 998 docker-registry:5000/my-jenkins-image:2.387.2-lts

docker ps -a

웹 검색 http://<Jenkins\_서버\_IP>:8080





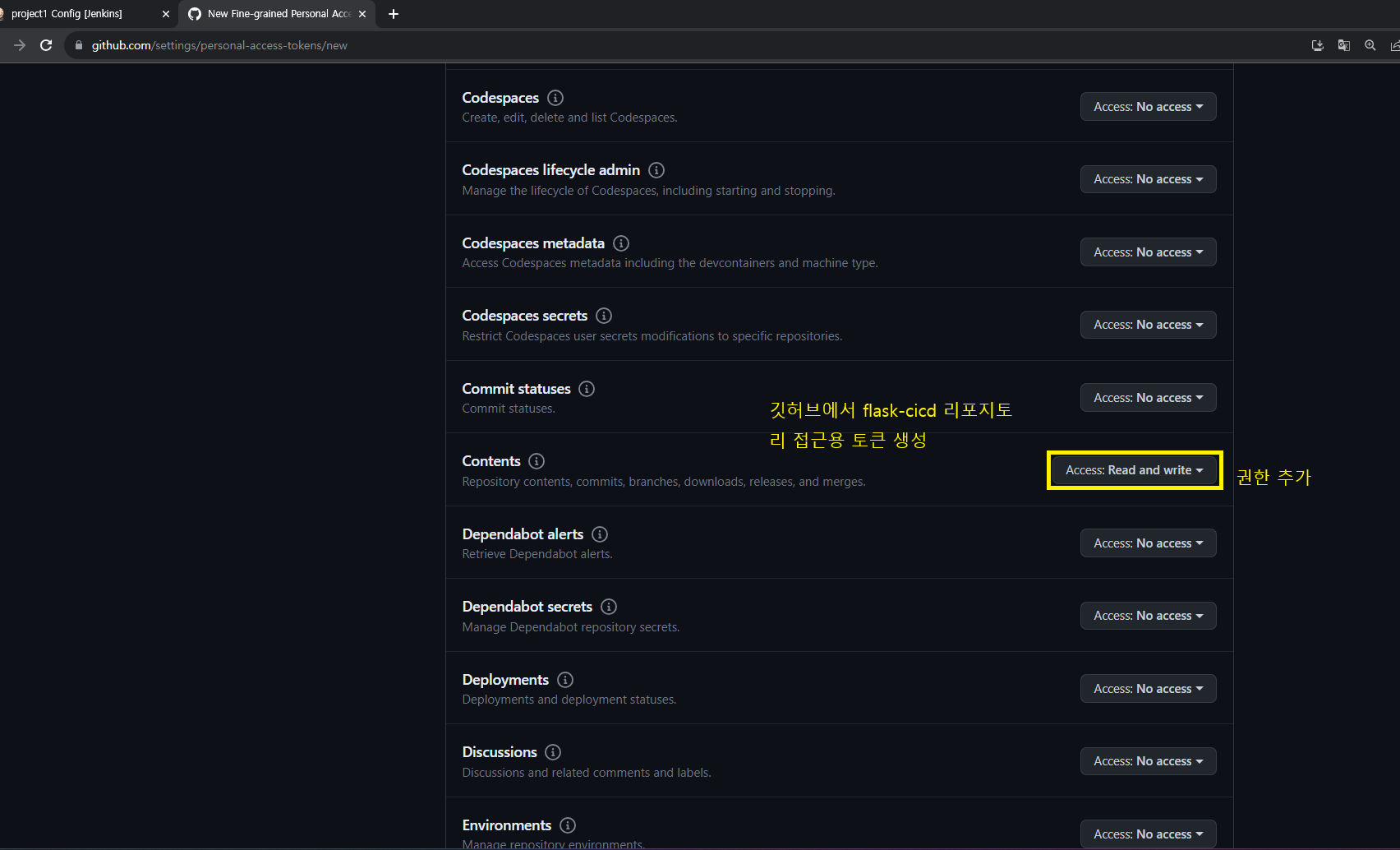


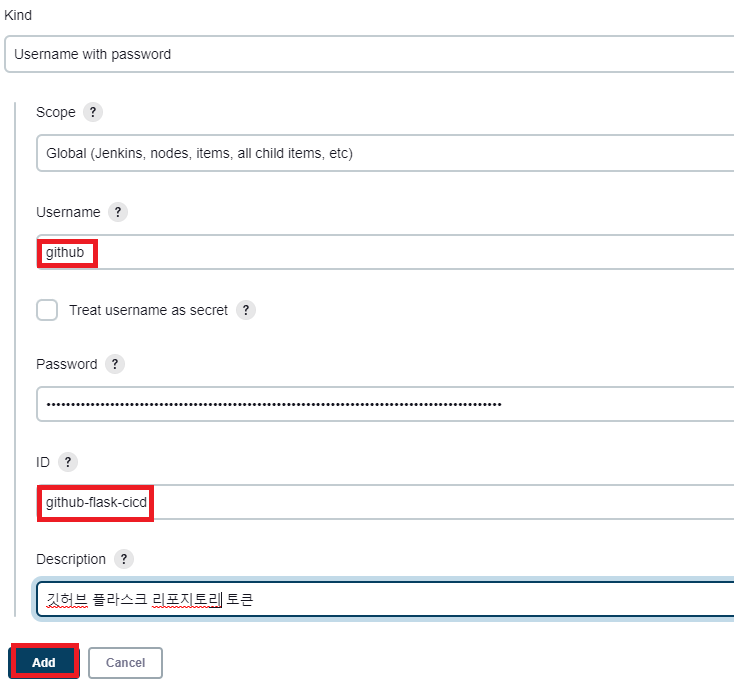
# 

# 

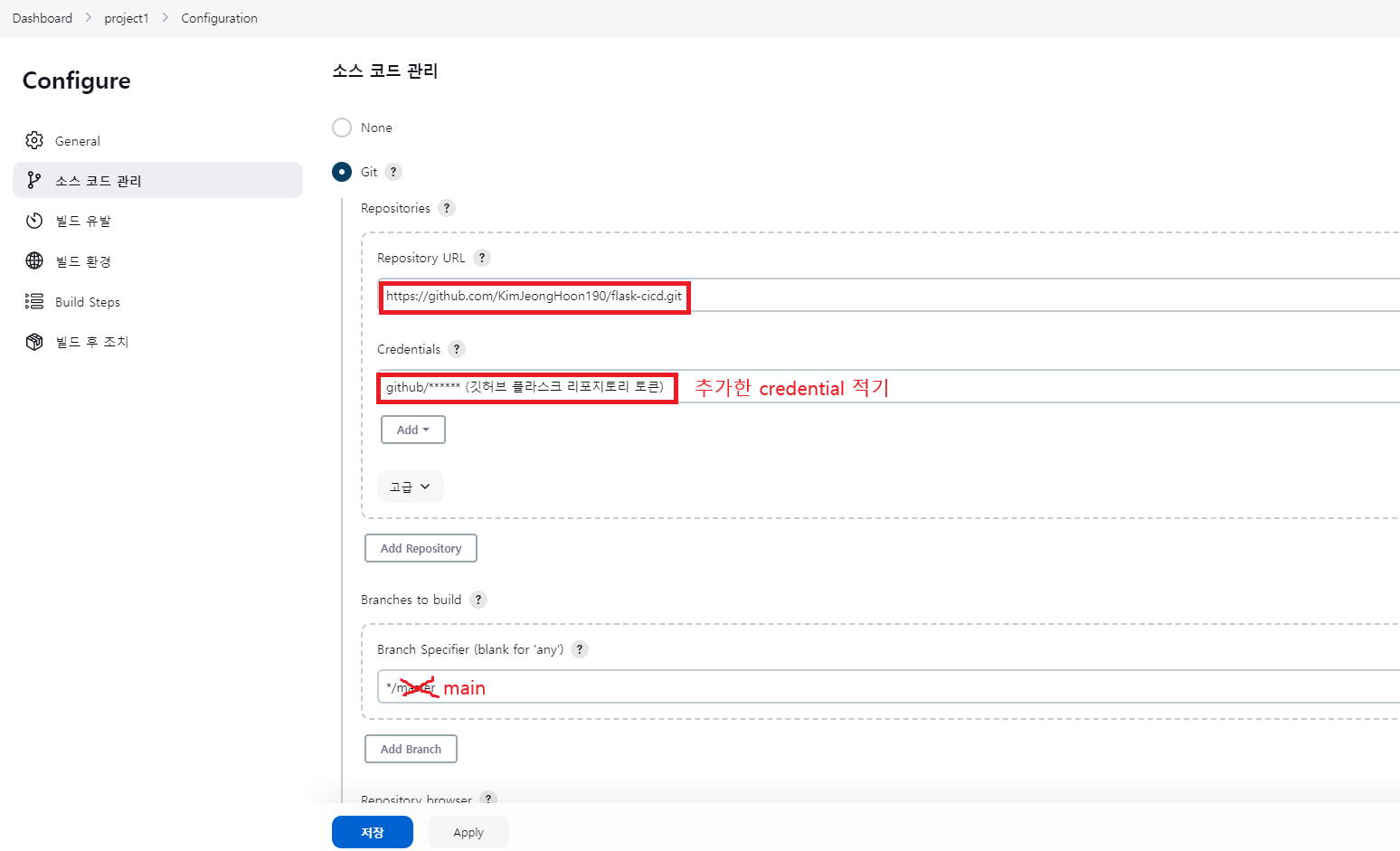
# 

+ 추가로 SonarQube Scanner Plugin도 설치





flask-cicd 실습 리포지토리 접근 및 수정용 토큰을 여기 credential에 복사 + 붙여넣기 해야 Jenkins에서 접근이 가능하다.



# 

# 

+

**SonarQube에서 Security란에서 SonarQube 접근용 토큰 또한 생성하여 비밀번호를 Jenkins의 credential에 복사 + 붙여넣기 하기.**

# NFS 서버

## 1. NFS 서버에서 NFS 서비스 설치

**NFS 서비스 설치**

sudo apt-get update

sudo apt-get install nfs-kernel-server

**NFS 서버 내 마운트 포인트 만들기**

sudo mkdir /nfs\_jenkins

sudo mkdir /nfs\_docker\_registry

**마운트 포인트 소유권 변경**

sudo chown nobody:nogroup /nfs\_jenkins

sudo chown nobody:nogroup /nfs\_docker\_registry

**외부 연결할 서버들 지정**

sudo vi /etc/exports



**파일 변경 반영 및 NFS 서비스 재시작**

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

## Jenkins 서버

**NFS 클라이언트 서비스 설치**

sudo apt-get install nfs-common

**마운트 포인트 만들기 + 연결**

sudo mkdir -p /mnt/nfs\_jenkins

sudo mount -t nfs <NFS\_서버\_IP>:/nfs\_jenkins /mnt/nfs\_jenkins

**볼륨을 NFS서버와 원격 연결 지정**

docker volume create --driver local --opt type=nfs --opt o=addr=<NFS\_서버\_IP>,rw --opt device=:/nfs\_jenkins jenkins-volume

**컨테이너 실행**

docker run -it -d -p 8080:8080 --restart=always --name jenkins -v jenkins-volume:/var/jenkins\_home/ -v /var/run/docker.sock:/var/run/docker.sock -v $(which docker):/usr/bin/docker --group-add 998 <젠킨스 이미지명>

## Docker-registry 서버

앞서 2번 Jenkins 서버에서 진행한 항목을 IP, 마운트 포인트 이름 등을 기호에 맞게 변경해서 진행할 수 있다.

# Ngrok 서버

**Curl 커맨드 설치**

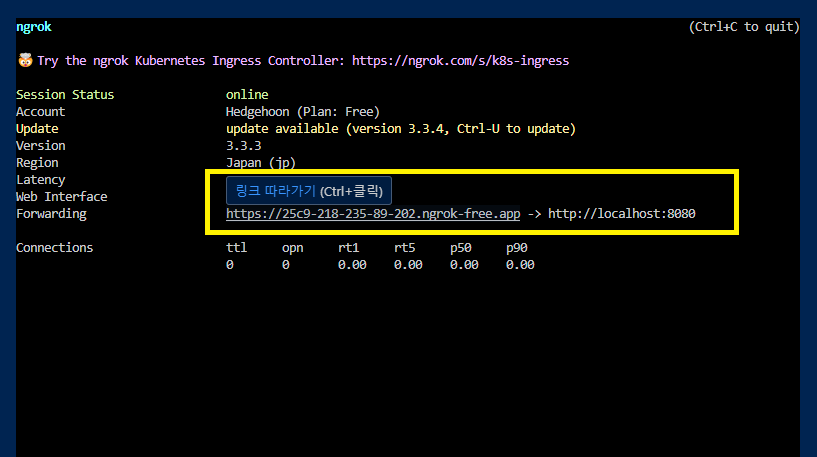
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null && echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | sudo tee /etc/apt/sources.list.d/ngrok.list && sudo apt update && sudo apt install ngrok

**Auth-token 삽입 (사전에 Ngrok 홈페이지 가입 후 발급받을 수 있다)**

ngrok config add-authtoken <Your\_Auth\_Token>

**URL 생성 및 접근**

ngrok http <Jenkins\_서버\_IP>:8080



해당 링크를 따라가서 Jenkins를 다시 사용해보자.

# SonarQube 서버 설치

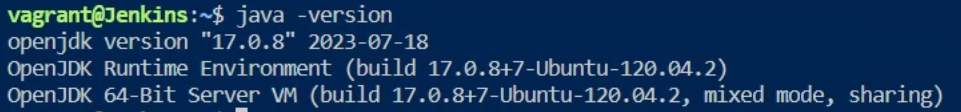
## 1. Java 17 설치

**sudo apt update**

**sudo apt install openjdk-17-jdk**

**java -version**

# 설치 완료 확인



2. SonarQube 다운로드 및 설치

**sudo curl -O https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip**

**sudo unzip sonarqube-9.9.0.65466.zip -d /opt/**

3. SonarQube 서비스 설정

**sudo tee /etc/systemd/system/sonarqube.service << EOF**

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/opt/sonarqube-9.9.0.65466/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube-9.9.0.65466/bin/linux-x86-64/sonar.sh stop

User=sonarqube

Group=sonarqube

Restart=always

[Install]

WantedBy=multi-user.target

EOF

4. SonarQube 사용자 및 그룹 생성

**sudo adduser --system --no-create-home --group sonarqube**

5. 권한 설정

**sudo chown -R sonarqube:sonarqube /opt/sonarqube-9.9.0.65466**

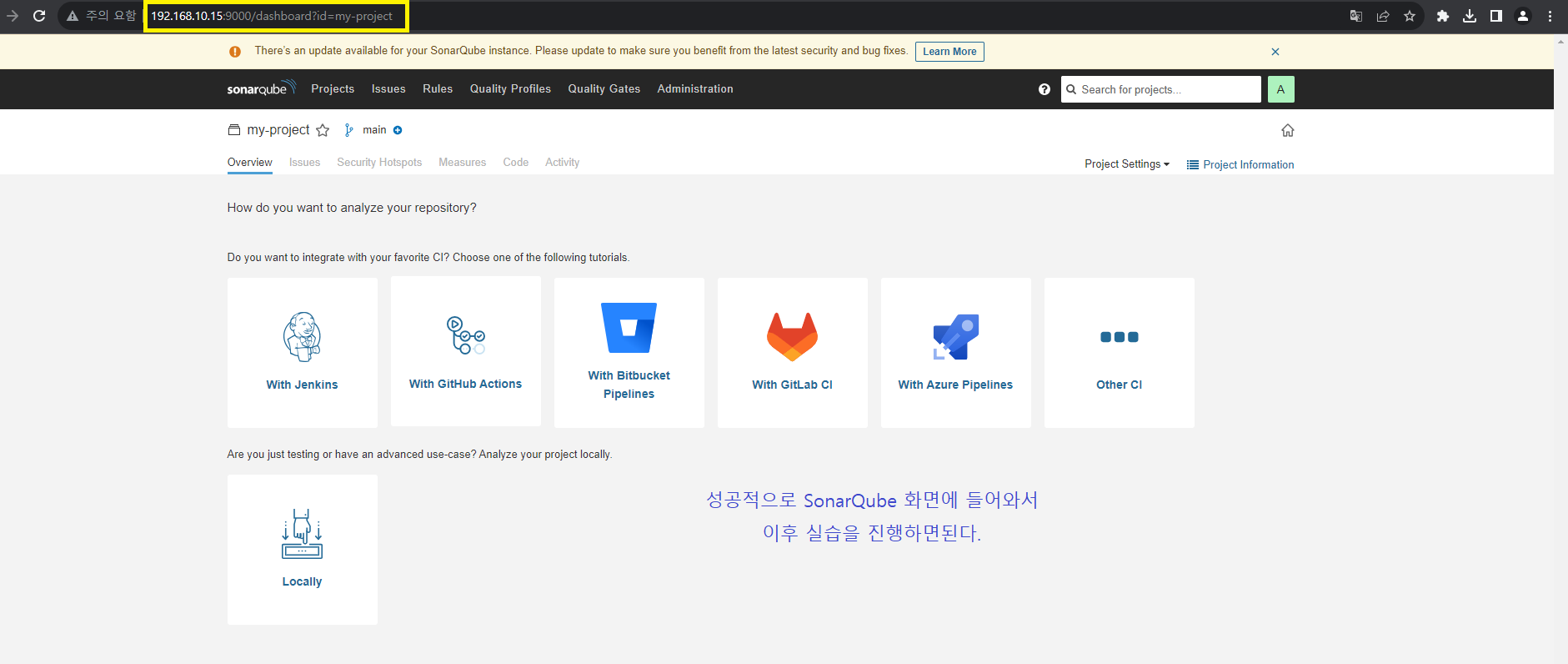
6. 서비스 시작 및 활성화

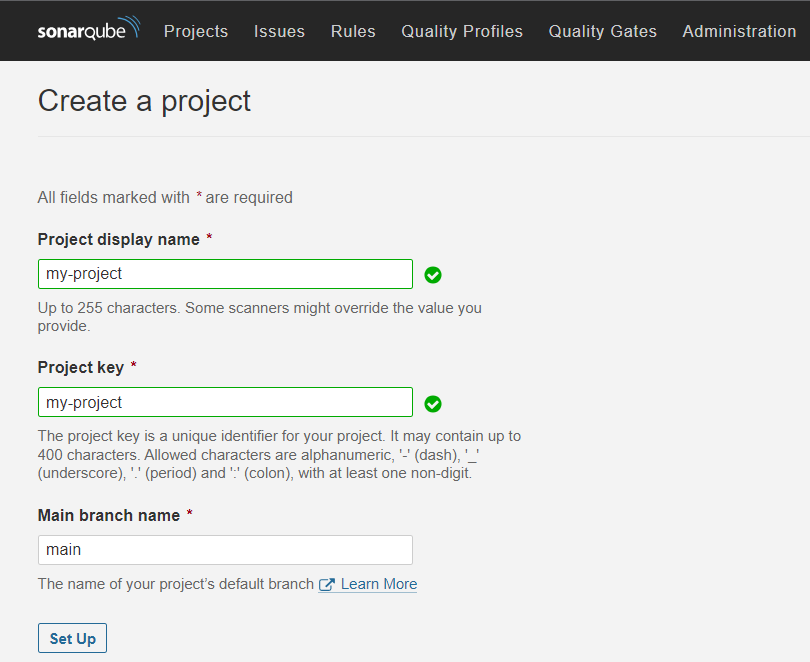
**sudo systemctl start sonarqube**

**sudo systemctl enable sonarqube**

7. SonarQube 웹 인터페이스 확인

**http://<Your\_SonarQube\_Server\_IP>:9000**

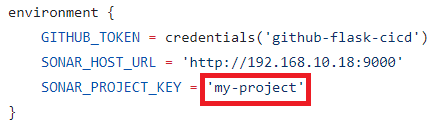
****

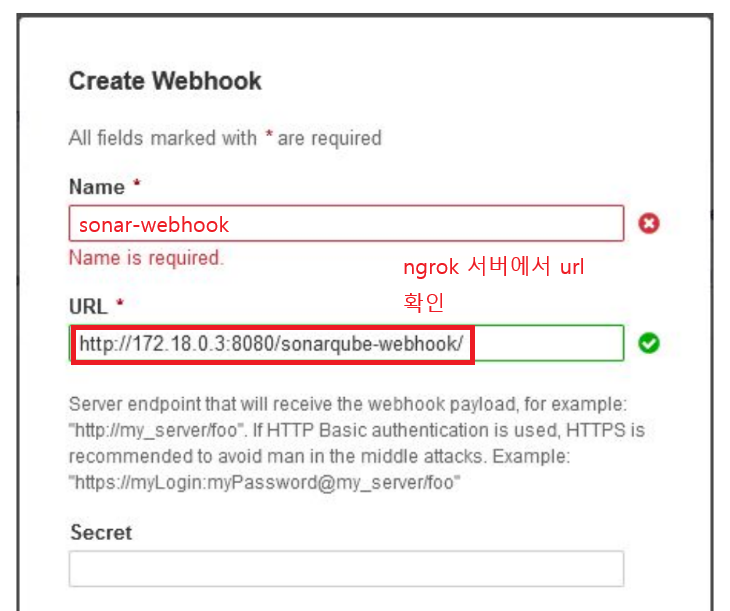


프로젝트 키 이름은 Jenkinsfile 에 적은 Project-key 네임과 일치시켜야 한다.

이렇게

Jenkinsfile >>





SonarQube 서버에서의 webhook 설정

--> Secret 부분이 Jenkins에서 설정한 SonarQube webhook의 secret과 일치해야 실습 진행이 가능하다.

