

설치 및 구성 메뉴얼

2023년 8월 19일

—

김정훈

프로젝트 Vagrantfile 구성

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

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"

  config.vm.define "dock" do |dock|
    dock.vm.hostname = "docker-registry"
    dock.vm.provider "virtualbox" do |vb|
      vb.name = "docker-registry"
      vb.cpus = 2
      vb.memory = 4096
    end
    dock.vm.network "private_network", ip: "192.168.10.14"
    dock.vm.provision "shell", inline: <<-SCRIPT
      sudo apt-get update -y
      sudo apt-get install -y ca-certificates curl gnupg lsb-release
      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
      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
      sudo apt-get update -y
      sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
      sudo usermod -a -G docker vagrant
      sudo systemctl enable docker
      sudo systemctl start docker
      docker login -u hedgehoon --password-stdin < /vagrant/env/docker_token
    SCRIPT
  end
end
```

```

config.vm.define "nfs" do |nfs|
  nfs.vm.hostname = "NFS"
  nfs.vm.provider "virtualbox" do |vb|
    vb.name = "NFS"
    vb.cpus = 2
    vb.memory = 2048
  end
  nfs.vm.network "private_network", ip: "192.168.10.16"
end

```

```

config.vm.define "jenk" do |jenk|
  jenk.vm.hostname = "Jenkins"
  jenk.vm.provider "virtualbox" do |vb|
    vb.name = "Jenkins"
    vb.cpus = 2
    vb.memory = 4096
  end
  jenk.vm.network "private_network", ip: "192.168.10.15"
  jenk.vm.provision "shell", inline: <<-SCRIPT
    sudo apt-get update -y
    sudo apt-get install -y ca-certificates curl gnupg lsb-release
    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
    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
    sudo apt-get update -y
    sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
    sudo usermod -a -G docker vagrant
    sudo systemctl enable docker
    sudo systemctl start docker
    docker login -u hedgehoon --password-stdin < /vagrant/env/docker_token
  SCRIPT
end

```

```
config.vm.define "ngrok" do |ngrok|
  ngrok.vm.hostname = "Ngrok-server"
  ngrok.vm.provider "virtualbox" do |vb|
    vb.name = "Ngrok-server"
    vb.cpus = 1
    vb.memory = 2048
  end
  ngrok.vm.network "private_network", ip: "192.168.10.17"
end

config.vm.define "sonar" do |sonar|
  sonar.vm.hostname = "SonarQube"
  sonar.vm.provider "virtualbox" do |vb|
    vb.name = "SonarQube"
    vb.cpus = 2
    vb.memory = 4096
  end
  sonar.vm.network "private_network", ip: "192.168.10.18"
end

end
```

파이프라인 작업 리포지토리

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

Jenkinsfile 구성

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

Docker-registry 설치

1. Docker 설치

Docker-registry 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
```

2. Docker-registry 서버 기능 구현

Docker-registry 서버에서 >>

```
docker pull registry:latest
```

#docker-registry 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

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password


```
docker exec -t jenkins /bin/bash -c "cat /var/jenkins_home/secrets/initialAdminPassword"docker exec -t  
jenkins /bin/bash -c "cat /var/jenkins_home/secrets/initialAdminPassword"
```

명령어로 나오는 초기 젠킨스 비밀번호를 Administrator password에 복사 + 붙여넣기 한다.

Continue

Getting Started

Create First Admin User

계정명

admin

암호

.....

암호 확인

.....

이름

admin

이메일 주소

hedgehoon@gmail.com

계정명, 암호, 이름 등을 기억하기 쉬운 것으로
지정하고 사용자 이메일 계정을 입력한 뒤 다음
화면으로 넘어간다.

Jenkins 2.387.2

[Skip and continue as admin](#)

[Save and Continue](#)

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.387.2

Not now

클릭

[Save and Finish](#)

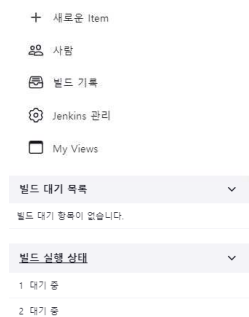
Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

클릭



Jenkins Dashboard 화면이 뜨고 정상적으로
설치가 된 것을 확인할 수 있다.

Jenkins에 오신 것을 환영합니다.

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

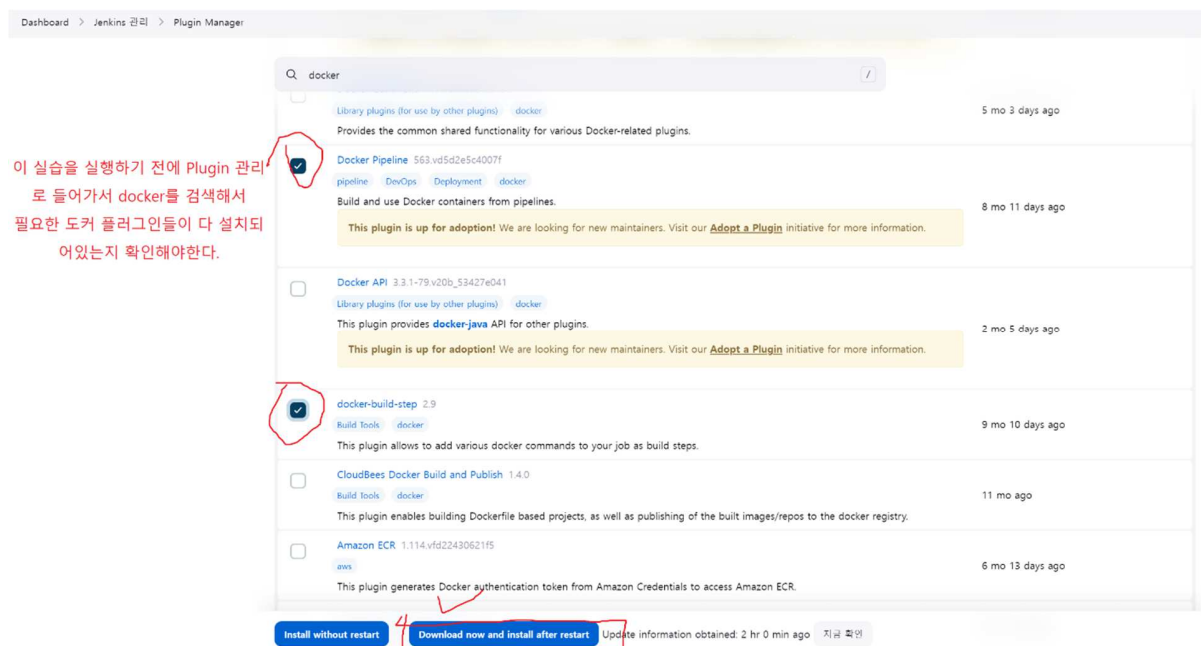
Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗



Dashboard > Jenkins 관리 > Plugin Manager

Updates

Available plugins

Installed plugins

Advanced settings

Plugins

utility

Install	Name ↓	Released
<input checked="" type="checkbox"/>	Pipeline Utility Steps 2.16.0 pipeline Build Tools Miscellaneous Utility steps for pipeline jobs.	1 mo 3 days ago
<input type="checkbox"/>	OWASP Dependency-Check 5.4.0 Security DevOps Build Tools Build Reports This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.	4 mo 23 days ago
<input type="checkbox"/>	Build Step Environment Filter Utilities 1.1 Utility classes useful for plugins implementing environment filters.	2 yr 3 mo ago

Install without restart
 Download now and install after restart
 Update information obtained: 22 min ago
 지금 확인

+ 추가로 SonarQube Scanner Plugin도 설치

project1 Config [Jenkins] x New Fine-grained Personal Acc. x

github.com/settings/personal-access-tokens/new

Codespaces ⓘ Create, edit, delete and list Codespaces.	Access: No access ▼
Codespaces lifecycle admin ⓘ Manage the lifecycle of Codespaces, including starting and stopping.	Access: No access ▼
Codespaces metadata ⓘ Access Codespaces metadata including the devcontainers and machine type.	Access: No access ▼
Codespaces secrets ⓘ Restrict Codespaces user secrets modifications to specific repositories.	Access: No access ▼
Commit statuses ⓘ Commit statuses.	Access: No access ▼
Contents ⓘ Repository contents, commits, branches, downloads, releases, and merges.	Access: Read and write ▼
Dependabot alerts ⓘ Retrieve Dependabot alerts.	Access: No access ▼
Dependabot secrets ⓘ Manage Dependabot repository secrets.	Access: No access ▼
Deployments ⓘ Deployments and deployment statuses.	Access: No access ▼
Discussions ⓘ Discussions and related comments and labels.	Access: No access ▼
Environments ⓘ Manage repository environments.	Access: No access ▼

깃허브에서 flask-cicd 리포지토리 접근용 토큰 생성

권한 추가

Kind

Username with password

Scope ?

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

Username ?

github

☐ Treat username as secret ?

Password ?

.....

ID ?

github-flask-cicd

Description ?

깃허브 플라스크 리포지토리 토큰

Add Cancel

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

Dashboard > project1 > Configuration

Configure

- General
- 소스 코드 관리**
- 빌드 유발
- 빌드 환경
- Build Steps
- 빌드 후 조치

소스 코드 관리

☐ None ☒ Git ?

Repositories ?

Repository URL ?
`https://github.com/KimJeongHoon190/flask-cicd.git`

Credentials ?
`github/***** (깃허브 플라스크 리포지토리의 토큰)` **추가한 credential 적기**

Branches to build ?

Branch Specifier (blank for 'any') ?
`*/main`

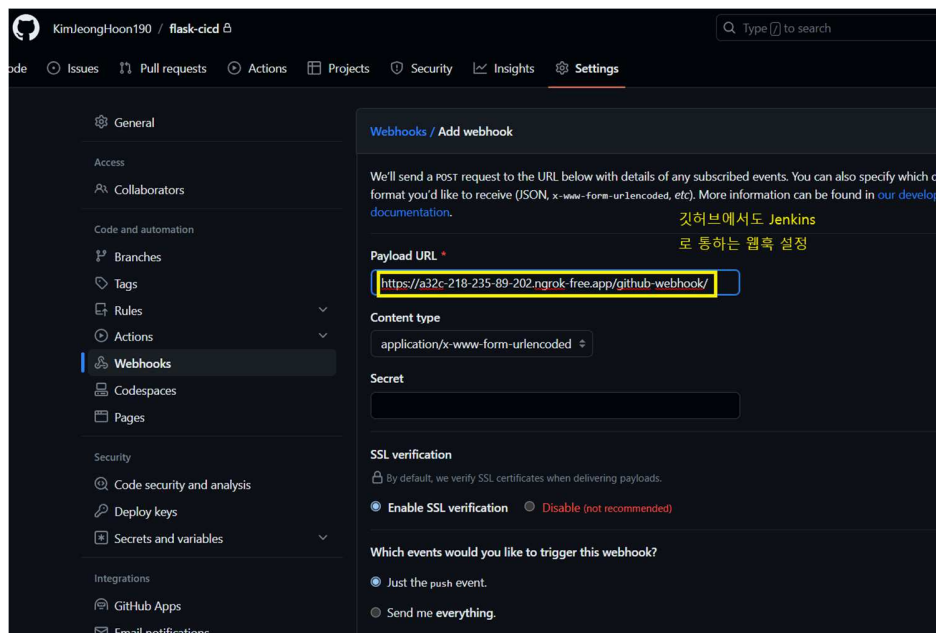
Repository browser ?

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

빌드 환경

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?



+

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
```

```
/nfs_jenkins 192.168.10.15(rw,sync,no_subtree_check,no_root_squash)  
/nfs_docker_registry 192.168.10.14(rw,sync,no_subtree_check,no_root_squash)
```

파일 변경 반영 및 NFS 서비스 재시작

```
sudo exportfs -a
```

```
sudo systemctl restart nfs-kernel-server
```

2. 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>
```

3. 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

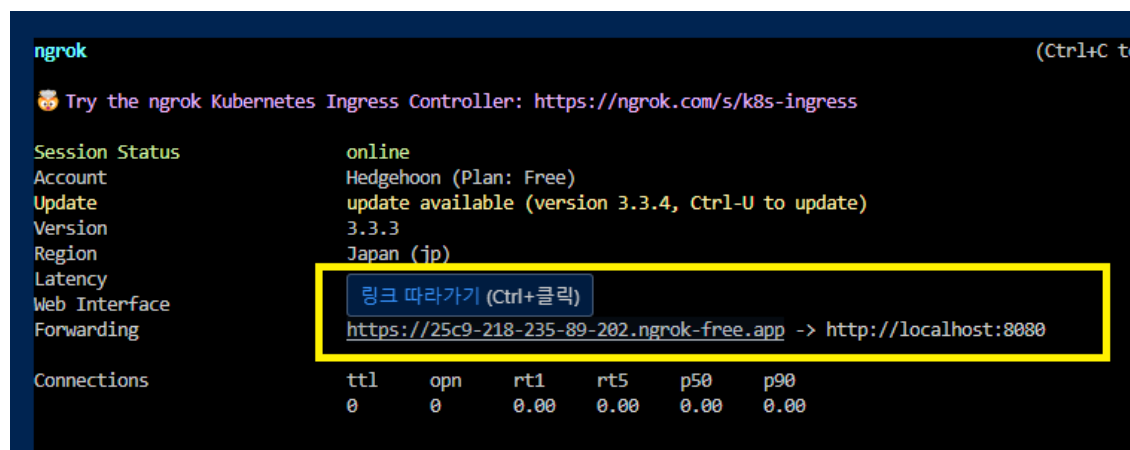
```
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo tee
/etc/apt/trusted.gpg.d/ngrok.asc >/dev/null && echo "deb https://ngrok-
agent.s3.amazonaws.com buster main" | sudo tee /etc/apt/sources.list.d/ngrok.list &&
sudo apt update && sudo apt install ngrok
```

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
```

```
vagrant@Jenkins:~$ java -version
openjdk version "17.0.8" 2023-07-18
OpenJDK Runtime Environment (build 17.0.8+7-Ubuntu-120.04.2)
OpenJDK 64-Bit Server VM (build 17.0.8+7-Ubuntu-120.04.2, mixed mode, sharing)
```

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

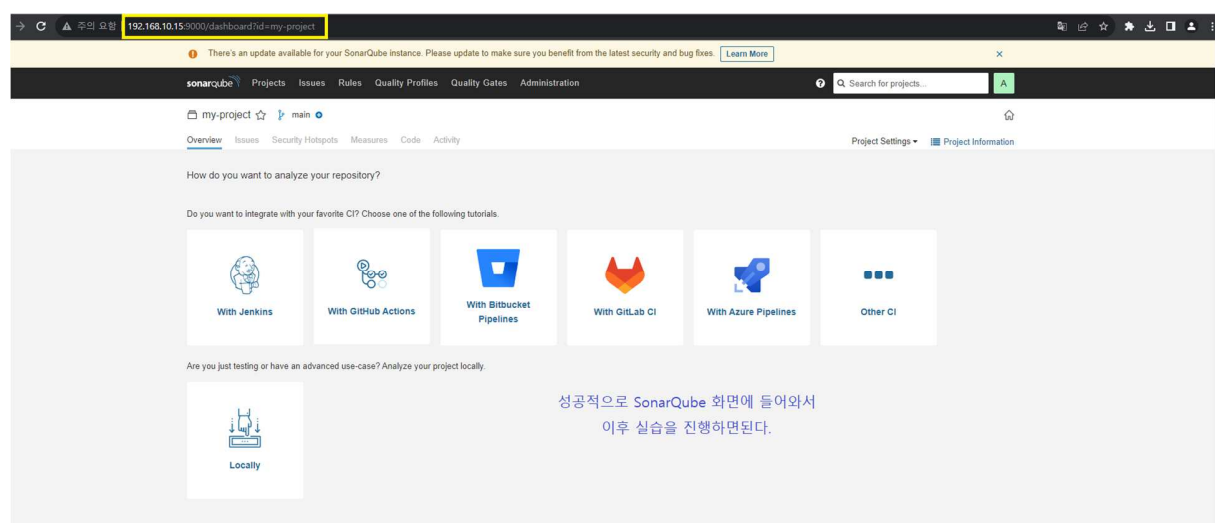
```
sudo systemctl start sonarqube
```

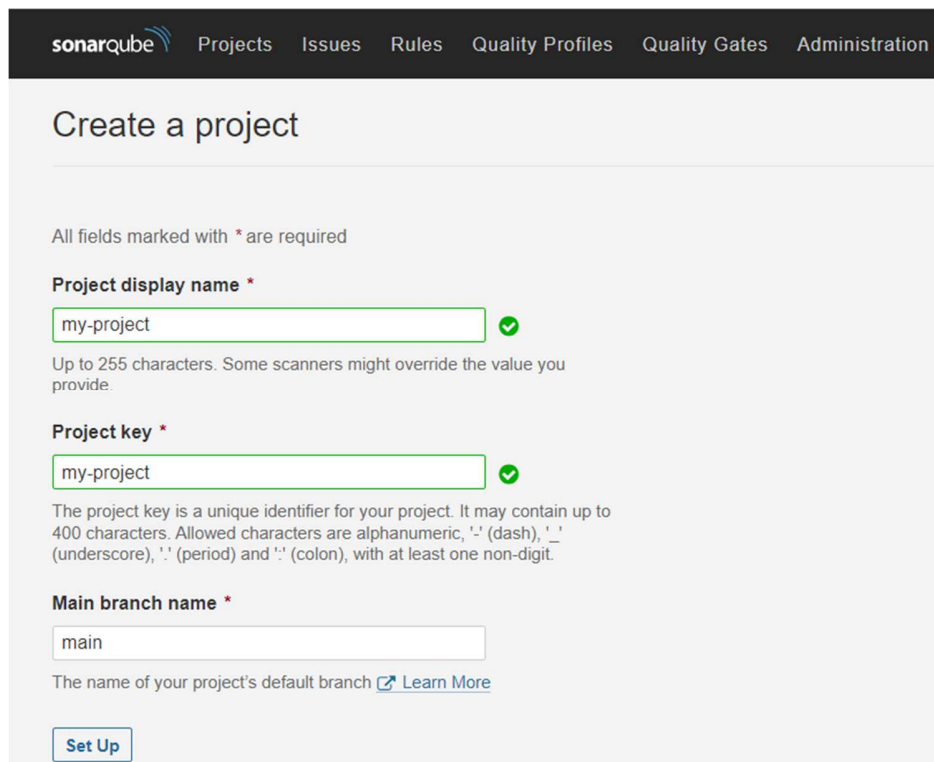


```
sudo systemctl enable sonarqube
```

7. SonarQube 웹 인터페이스 확인

http://<Your_SonarQube_Server_IP>:9000





sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Create a project

All fields marked with * are required

Project display name *

my-project ✓

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

Project key *

my-project ✓

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 *

main

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

[Set Up](#)

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

이렇게

Jenkinsfile >>

```
environment {  
    GITHUB_TOKEN = credentials('github-flask-cicd')  
    SONAR_HOST_URL = 'http://192.168.10.18:9000'  
    SONAR_PROJECT_KEY = 'my-project'  
}
```

Create Webhook

All fields marked with * are required

Name *

sonar-webhook ✖

Name is required. ngrok 서버에서 url 확인

URL *

http://172.18.0.3:8080/sonarqube-webhook/ ✔

Server endpoint that will receive the webhook payload, for example:
 "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example:
 "https://myLogin:myPassword@my_server/foo"

Secret

SonarQube 서버에서의 webhook 설정

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

→ 주의 요함 | 192.168.10.18:9000/project/webhooks?id=my-project

There's an update available for your SonarQube instance. Please update to make sure you benefit from the latest security and bug fixes. [Learn More](#)

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

my-project main

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).

Create

Name	URL	Secret?	Last delivery
sonar-webhook	http://192.168.10.15:9000/sonarqube-webhook/	Yes	Never

⚙️

