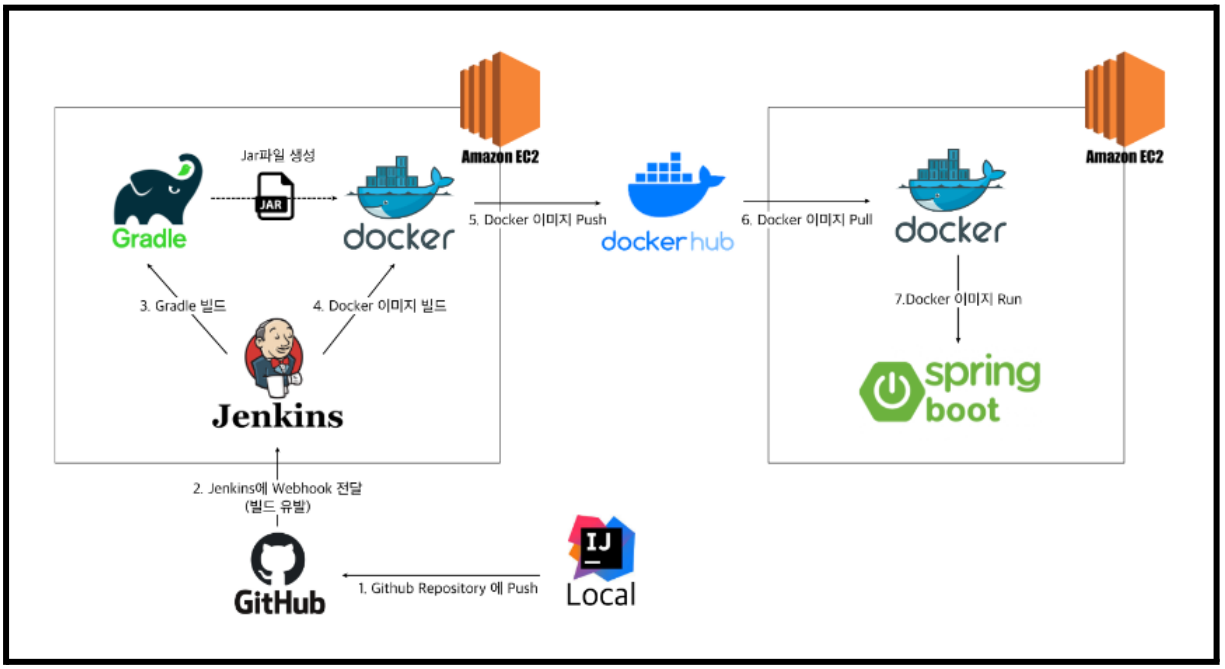
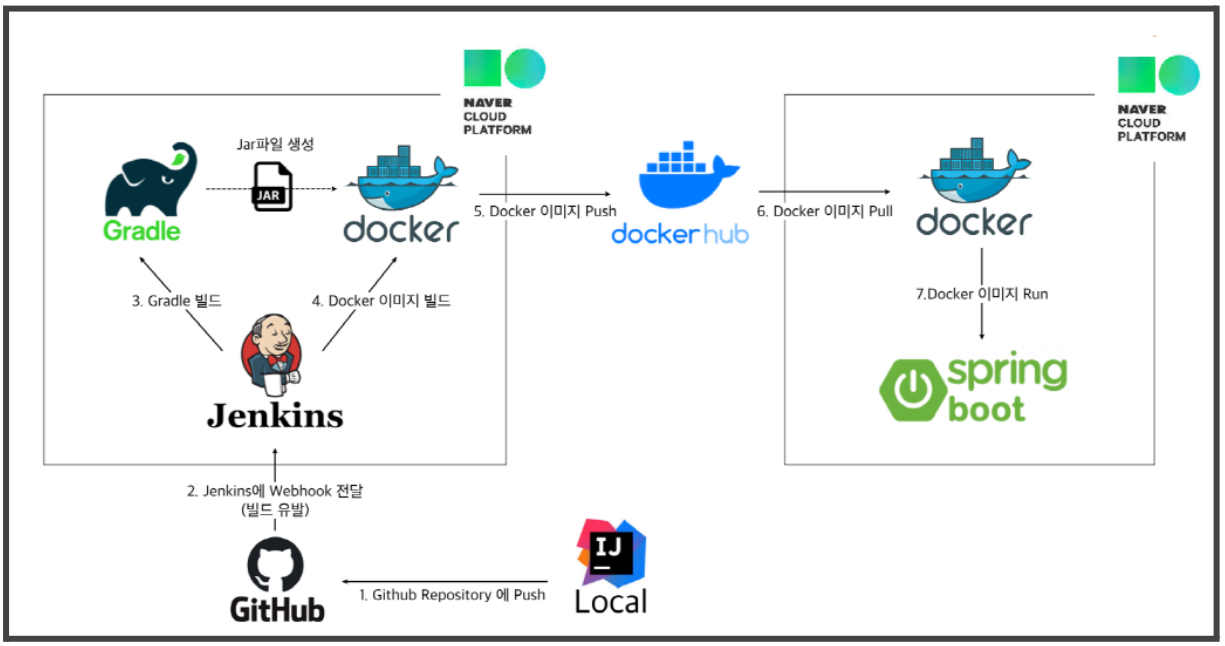
https://velog.io/@haeny01/AWS-Jenkins%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-Docker-x-SpringBoot-CICD-%EA%B5%AC%EC%B6%95





ncloud 에 새로운 server 2개 생성하기(비번변경 bit!@#123)

1. react-jenkins

2. react-springboot

도커 설치 스크립트 다운로드

```
# apt-get update
# apt-get install curl
# curl https://get.docker.com > docker-install.sh
# chmod 755 docker-install.sh
```

도커 설치

```
# ./docker-install.sh
```

도커 버전 확인

```
# docker -v
```

도커 이미지 목록 확인하기-아직 아무거도 없음

```
docker image ls
docker images
```

Jenkins 사용법/Jenkins 설치

젠킨스 도커 컨테이너에서 사용할 브릿지 네트워크를 준비한다.

```
# docker network ls
# docker network create jenkins
# docker network ls
```

JDK 젠킨스 이미지 가져오기

```
# docker pull jenkins/jenkins:lts-jdk11  //11버전
# docker pull jenkins/jenkins:lts-jdk17  //17버전
# docker image ls
```

```
#apt-get install vim  (vi 에디터 인스톨)
```

작업 디렉토리 생성

mkdir jenkins
cd jenkins

install-docker.sh 파일 생성

vi install-docker.sh

install-docker.sh 파일 내용

```
apt-get update

apt-get -y install apt-transport-https \
     apt-utils \
     ca-certificates \
     curl \
     gnupg2 \
     zip \
     unzip \
     acl \
     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
```

도커 빌드 파일 생성

vi Dockerfile

Dockerfile 내용

```
FROM jenkins/jenkins:lts-jdk11

USER root

COPY install-docker.sh /install-docker.sh
RUN chmod +x /install-docker.sh
RUN /install-docker.sh

RUN usermod -aG docker jenkins
RUN setfacl -Rm d:g:docker:rwx,g:docker:rwx /var/run/

USER jenkins
```

도커 이미지 생성

```
# docker build -t leemoonhee/react-docker:1.0 .
```

도커 이미지를 도커 허브 사이트에 업로드 하기

```
# docker login
# docker push leemoonhee/react-docker:1.0
```

도커허브에서 확인

컨테이너 생성 및 실행하기(DooD -Docker Out of Docker 방식)

```
docker run --privileged -d -v /var/run/docker.sock:/var/run/docker.sock -v
jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000
```

--restart=on-failure --network="jenkins" --name docker-jenkins leemoonhee/react-docker:1.0

docker container ls

볼륨 확인

docker volume ls

혹시 볼륨을 지워야할 상황이 생겼을 경우
docker volume rm jenkins_home

---

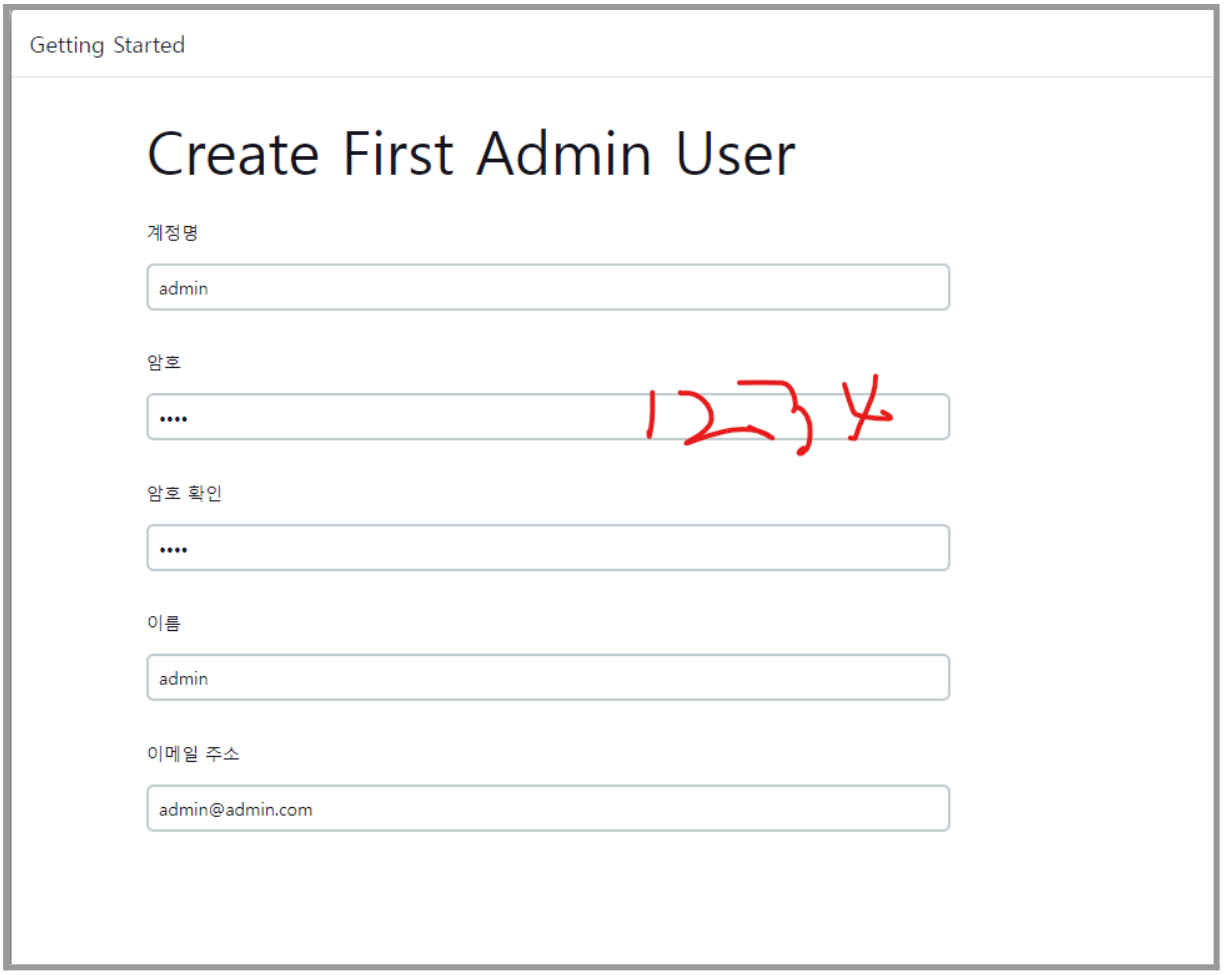
jenkins 접속후 암호찾아서 넣기

각자 자기 서버의 젠키스에 접속
http://223.130.163.226:8080

초기 비밀번호 확인 방법

docker logs docker-jenkins

암호 찾아서 넣고 install - Getting Started 진행됨…



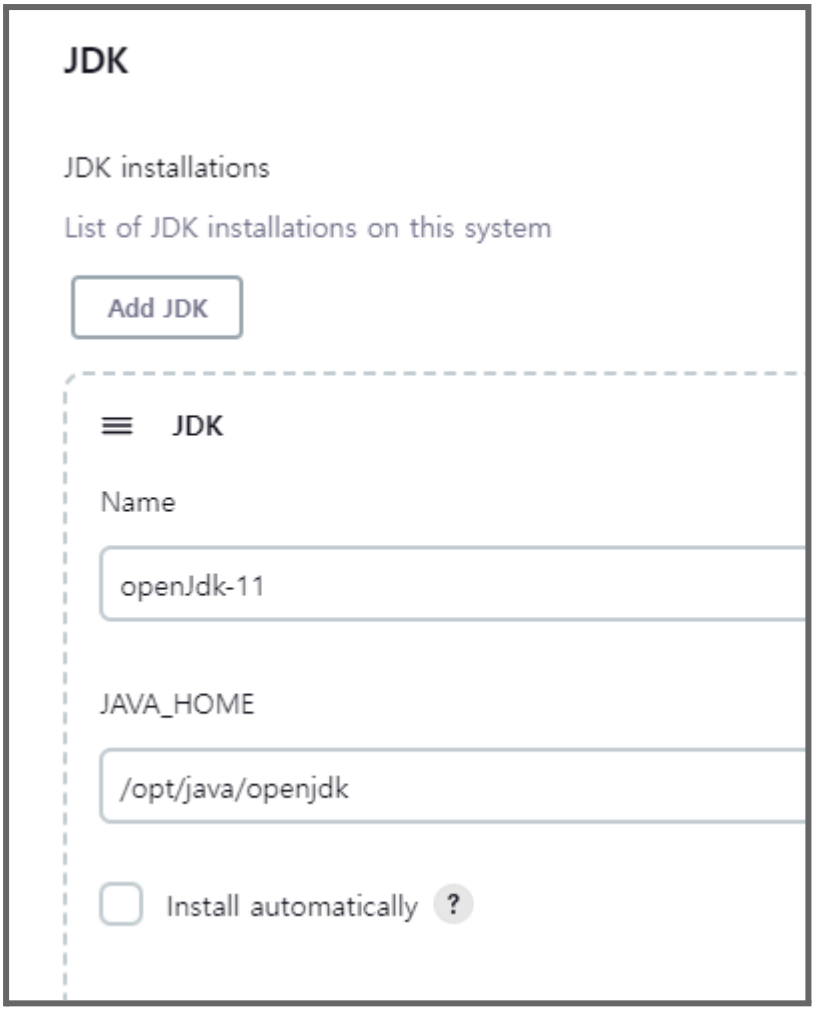
Jenkins 환경설정

JDK 및 Gradle 설정 - tools

JDK - 11 또는 17 선택

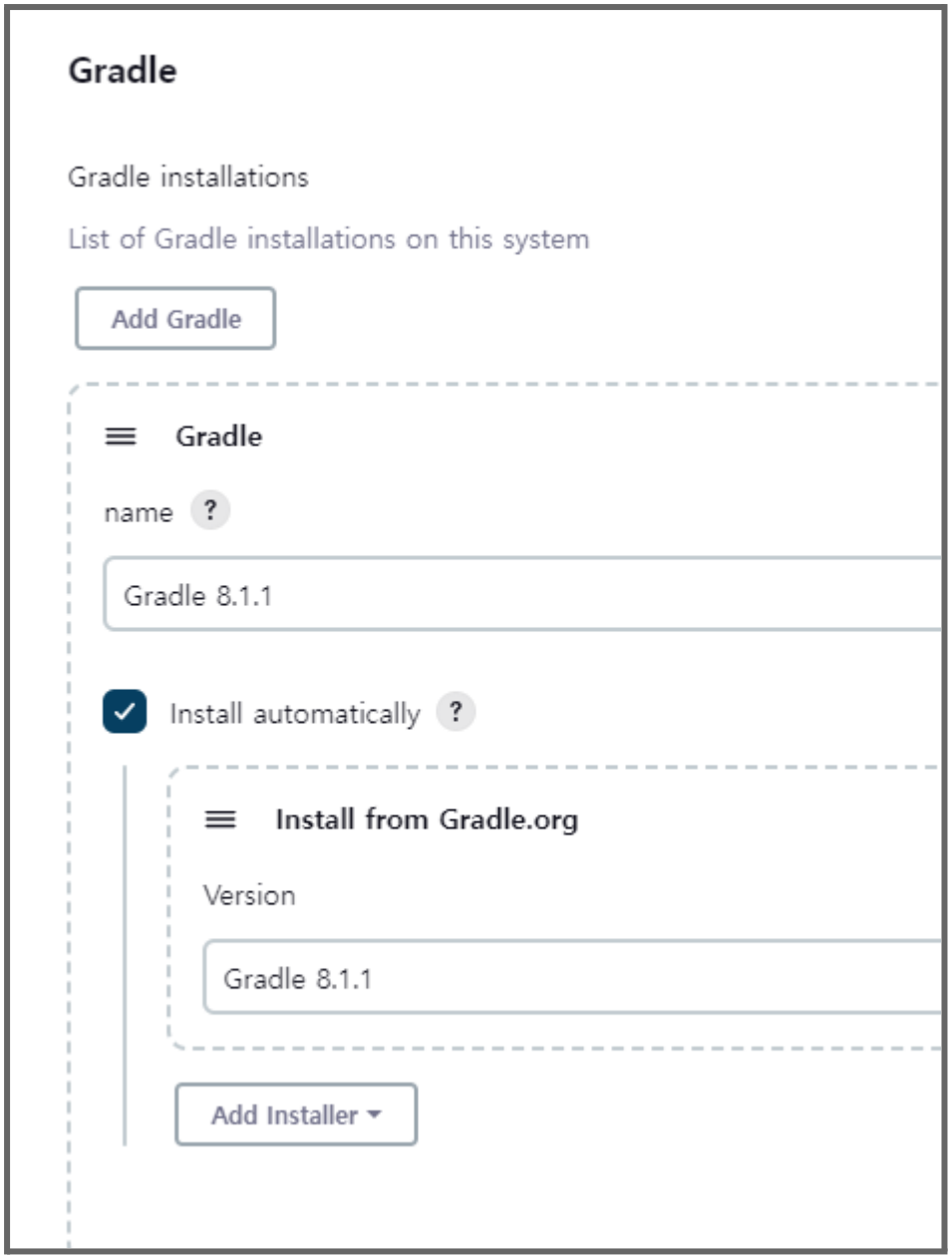
NAME : openJdk-11
JAVA_HOME : /opt/java/openjdk
Apply

(docker inspect docker-jenkins 에서 확인 가능)

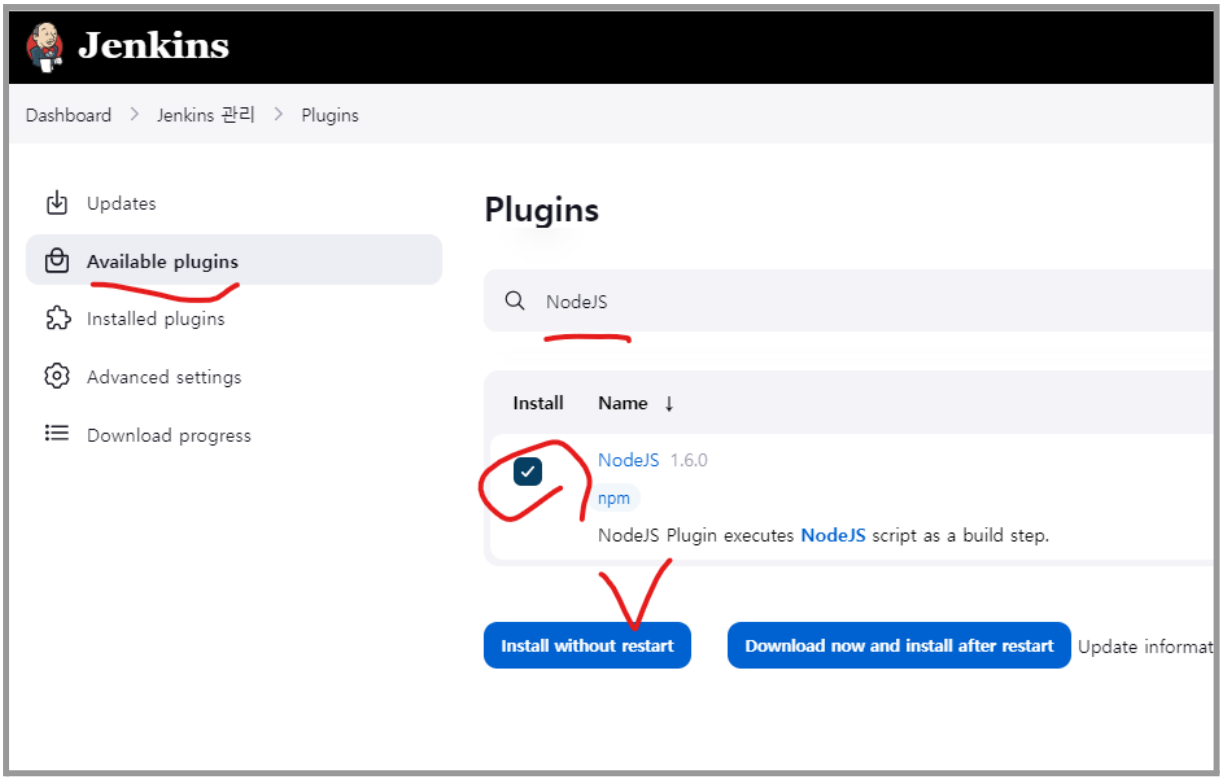


Gradle

Add Gradle 선택
  Name: Gradle 8.1.1
  Install automatically: 체크
  Version: 8.1.1 선택
  Apply 클릭



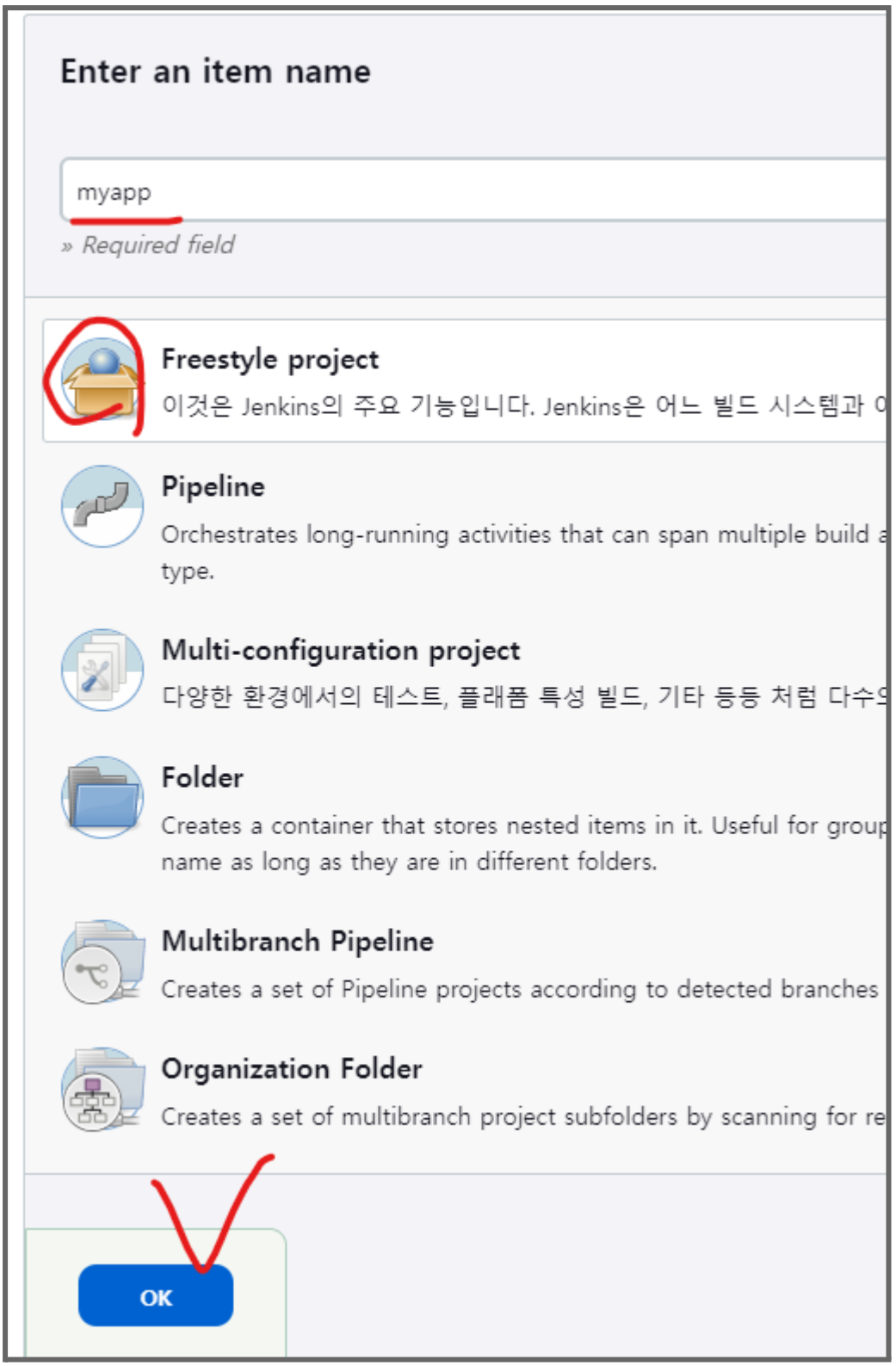
Node.js 설정 - jenkins관리 - 플러그인관리



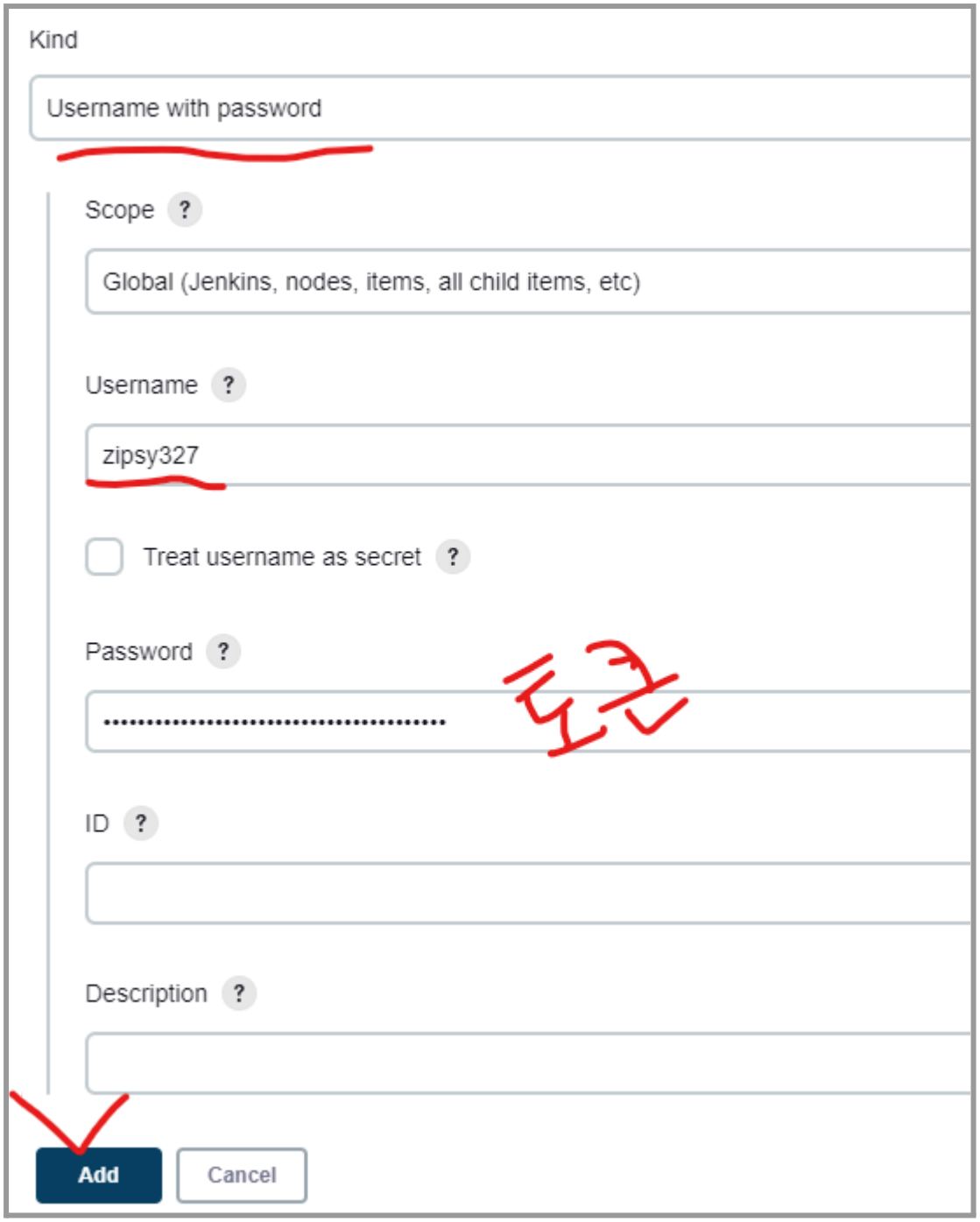
tools

Add NodeJS 클릭
Name: NodeJS 18.16.0
Install automatically: 체크
Version: 18.16.0 선택
Apply 클릭

프로젝트 생성
새로운Item-myapp-Freestyle-Ok

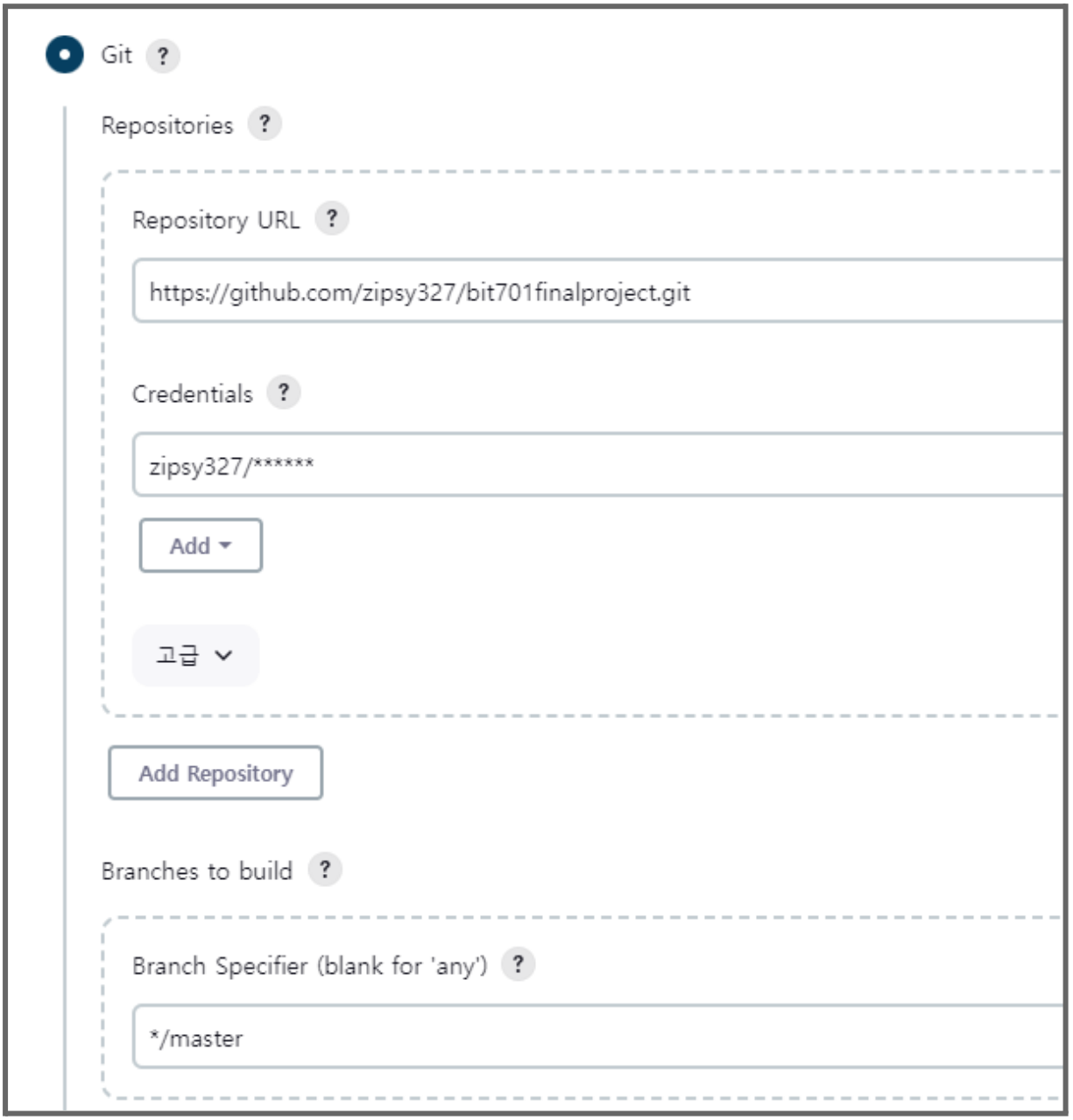

Enter an item name
myapp
» Required field
Freestyle project
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과
Pipeline
Orchestrates long-running activities that can span multiple build
type.
Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등 처럼 다수
Folder
Creates a container that stores nested items in it. Useful for group
name as long as they are in different folders.
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches
Organization Folder
Creates a set of multibranch project subfolders by scanning for re
OK

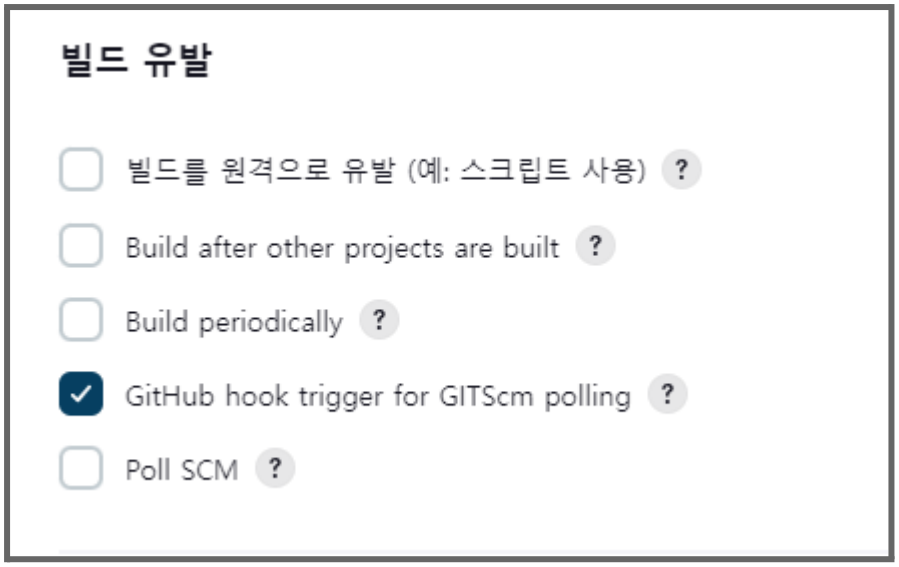

General
설명
react +springboot 배포
[Plain text] 미리보기
GitHub project
Project url
https://github.com/zipsy327/bit701finalproject.git

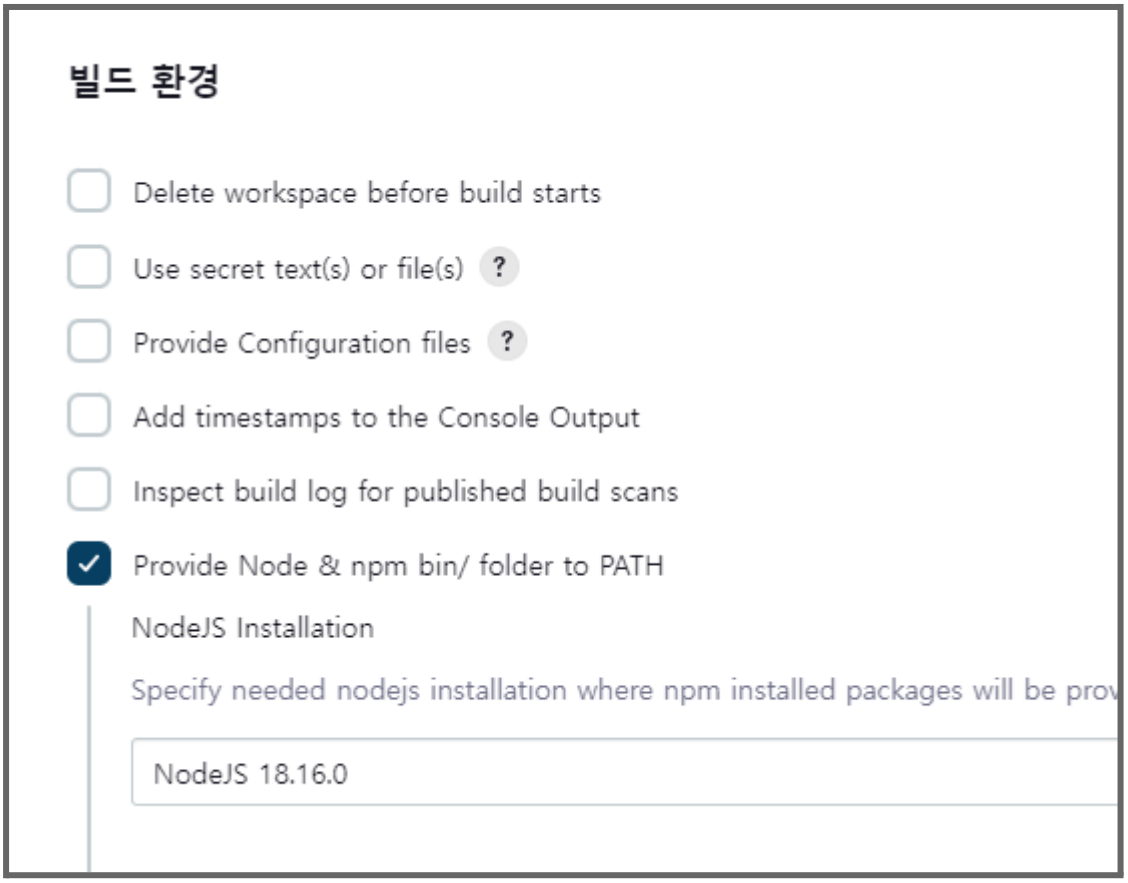


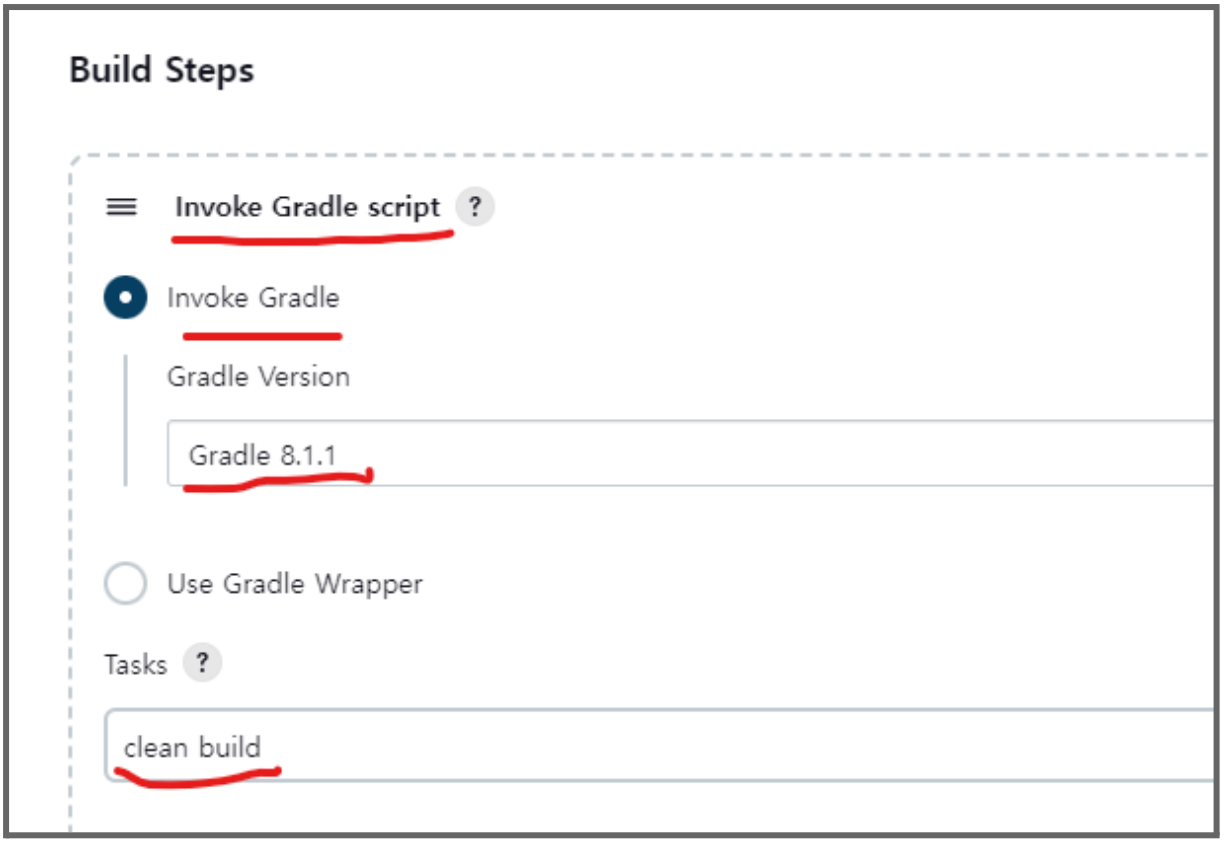
Credentials: (push를 할 경우)
Add 버튼 클릭: Add Jenkins 선택
Username with Password 선택
Username: 깃허브 사용자이름
Password: 깃허브 토큰
'Add' 클릭
username/토큰 선택

Branch Specifier: */master

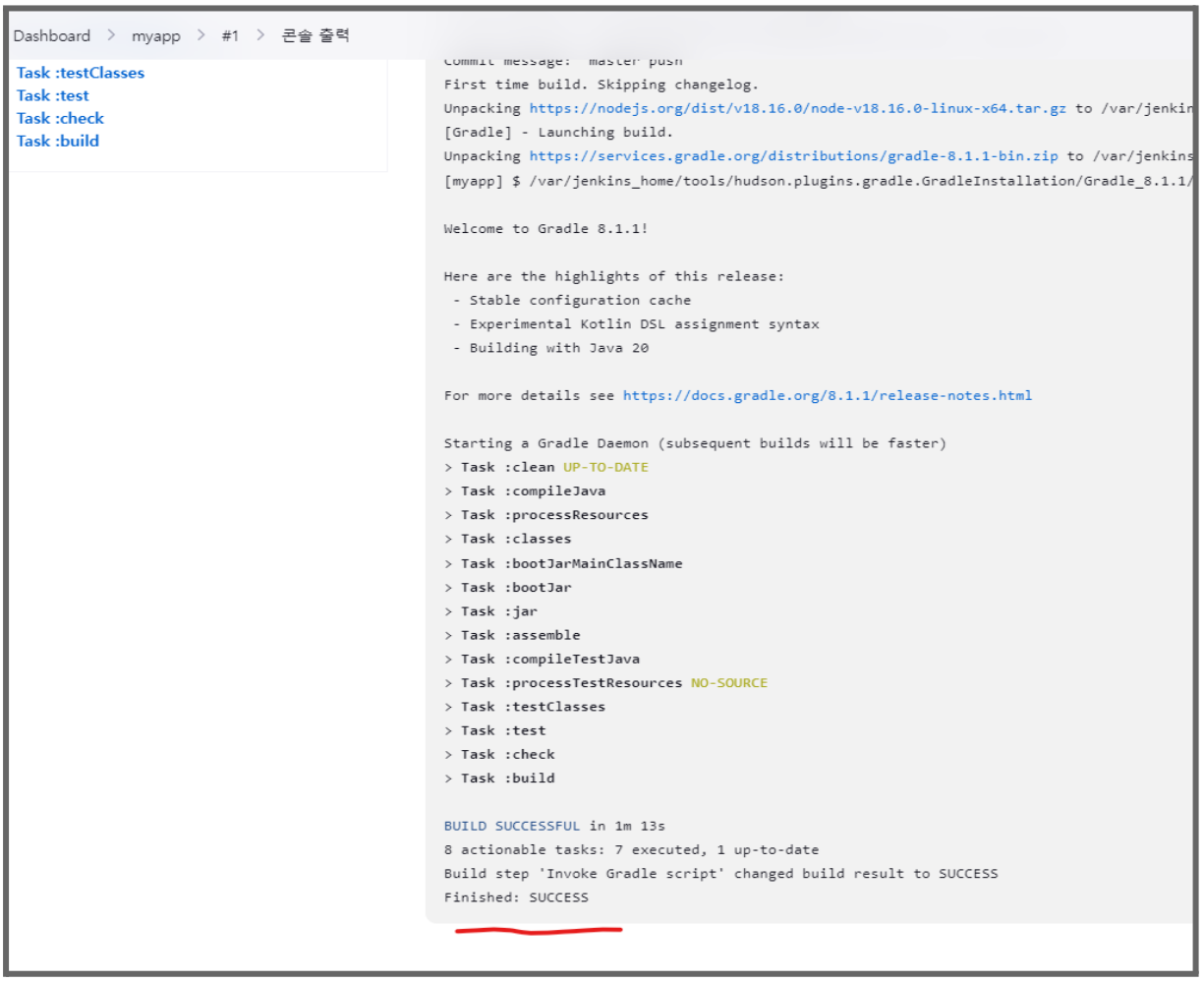



빌드 유발
빌드를 원격으로 유발 (예: 스크립트 사용)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM


빌드 환경
Delete workspace before build starts
Use secret text(s) or file(s)
Provide Configuration files
Add timestamps to the Console Output
Inspect build log for published build scans
Provide Node & npm bin/ folder to PATH
NodeJS Installation
Specify needed nodejs installation where npm installed packages will be prov
NodeJS 18.16.0


Build Steps
Invoke Gradle script
Invoke Gradle
Gradle Version
Gradle 8.1.1
Use Gradle Wrapper
Tasks
clean build

저장후 빌드하기, 빌드 성공후 확인



스프링프로젝트 내에 Dockerfile 생성

```
FROM openjdk:11

ARG JAR_FILE=build/libs/SpringReactProject-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT [ "java", "-jar", "app.jar" ]
```

---

build.gradle 은

https://github.com/zipsy327/PipeTestGradle.git

여기서 확인후 잘보고 없는 부분 복사

저장후 빌드하기, 빌드 성공후 확인

docker exec -itu 0 docker-jenkins bash 접속 (u:user 0 은 root)

cd /var/jenkins_home/workspace/myapp

cd /var/jenkins_home/workspace/myapp/src/main/resources/static
cat index.html

다른서버에 자동 배포하기

Repository/Settings/Webhooks
Add webook 클릭
Payload URL:
http://젠킨스서버주소:8080/github-webhook/
Content type: application/json
저장(Add Webhook)

스프링부트 애플리케이션 docker 이미지 생성 및 도커 허브에 push 하기

구성

Build Steps
Add build step : Execute shell 클릭
docker build -t [dockerHub UserName]/[dockerHub Repository]:[version] .
docker login -u '도커허브아이디' -p '도커허브비번' docker.io
docker push [dockerHub UserName]/[dockerHub Repository]:[version]

실제코드
docker build -t hub아이디/react-docker:2.0 .
docker login -u 'hub아이디' -p 'hub비번' docker.io

docker push hub아이디/react-docker:2.0

/var/run/docker.sock의 permission denied 발생하는 경우
호스트# chmod 666 /var/run/docker.sock

다시 빌드후 success 나오면

docker image 확인 : docker images

```
root@react-jenkins:~# docker images
REPOSITORY                TAG         IMAGE ID       CREATED              SIZE
leemoonhee/react-docker   2.0         e96d8600ebcf   About a minute ago   672MB
leemoonhee/react-docker   1.0         7ff86dcb34af   4 hours ago          1.04GB
jenkins/jenkins           lts-jdk11   c14d340748cf   3 weeks ago          471MB
root@react-jenkins:~#
```

스프링부트 애플리케이션 컨테이너 실행하기/ssh-key 등록)

jenkins 서버에 작업 디렉토리 만들기

mkdir springboot
cd springboot

---

ssh key 생성

root@react-jenkins:~/springboot# docker exec -itu 0 docker-jenkins bash
root@젠킨스컨테이너:/# ssh-keygen -t rsa -C "docker-jenkins-key" -m PEM -P "" -f /root/.ssh/docker-jenkins-key
root@젠킨스컨테이너:/# ls ~/.ssh/
docker-jenkins-key docker-jenkins-key.pub //이렇게 두개가 생겨있다. pub 는 공개키

root@젠킨스컨테이너:/# cat ~/.ssh/docker-jenkins-key : 안에 내용 확인하기
root@젠킨스컨테이너:/# cat ~/.ssh/docker-jenkins-key.pub : 안에 내용 확인하기

SSH-KEY 개인키 파일을 젠킨스 홈 폴더에 두기

root@젠킨스컨테이너:/# cp ~/.ssh/docker-jenkins-key /var/jenkins_home/
root@젠킨스컨테이너:/# chmod +r /var/jenkins_home/docker-jenkins-key

여기까지 작업후 빠져나오기 : exit

---

docker-jenkins 컨테이너의 SSH-KEY 공개키 파일을 Host로 복사해오기

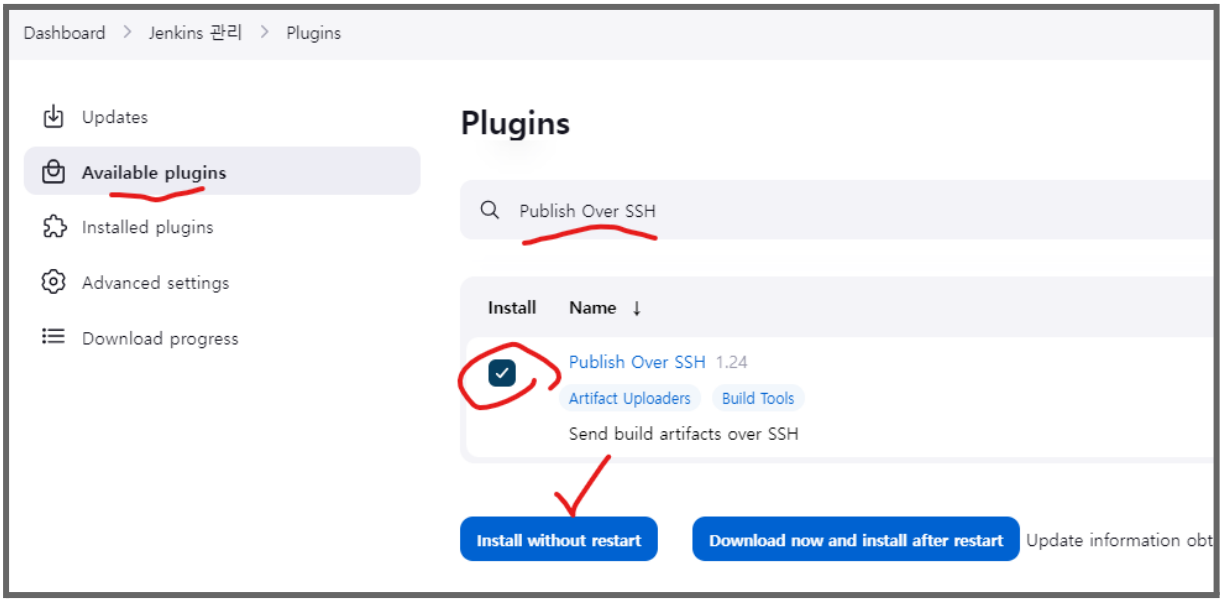
root@react-jenkins:~/springboot# docker cp docker-jenkins:/root/.ssh/docker-jenkins-key.pub ./docker-jenkins-key.pub
root@react-jenkins:~/springboot# cat docker-jenkins-key.pub

스프링부트 서버에 SSH-KEY 공개키 등록하기(react-springboot 서버로 가서 작업)

root@react-springboot:~# mkdir .ssh (ls -al 로 확인)
root@react-springboot:~# cd .ssh
root@react-springboot:~/.ssh# vi authorized_keys

Jenkins 컨테이너의 공개키 복사
(root@react-jenkins:~/springboot# cat docker-jenkins-key.pub)

Jenkins에 Publish Over SSH 플러그인 설정
플러그인 설치
Jenkins 관리
플러그인 설치
Available plugins 탭
Publish Over SSH 플러그인 설치



SSH 플러그인 연동

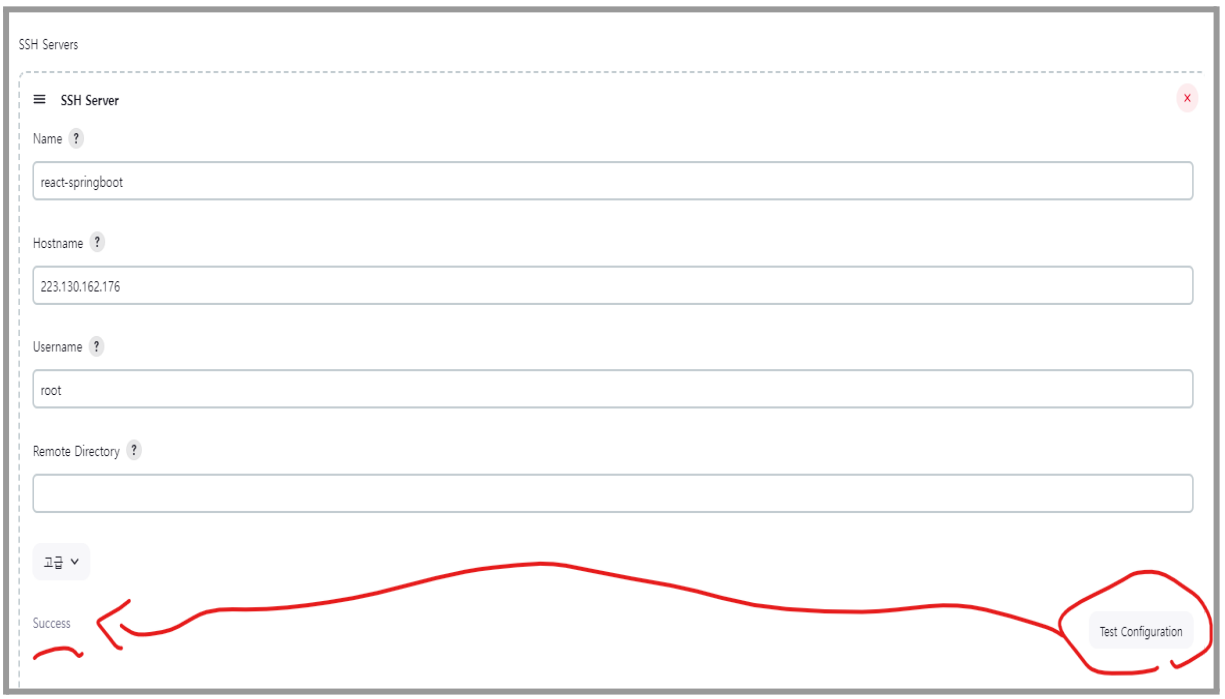
Jenkins 관리
    시스템 설정
        Publish over SSH
        Passphrase: 접속하려는 컨테이너의 root 암호 <=== 암호로 접속할 때
        Path to key: docker-jenkins-key <=== SSH-KEY로 접속할 때(private key 파일 경로, JENKINS_HOME 기준)
        Key: 개인키 파일의 내용 <=== SSH-KEY 개인키 파일의 내용을 직접 입력할 때

추가 버튼 클릭
    SSH Servers
        Name: 임의의서버 이름
        Hostname: 스프링부트서버의 IP 주소
        Username: 사용자 아이디
        Test Configuration 버튼 클릭
        Success OK!

private key 복사방법

root@react-jenkins:~/springboot# docker exec -itu 0 docker-jenkins bash
root@bd77b134919b:/# cat ~/.ssh/docker-jenkins-key

SSH Server 설정 (react-springboot 의 이름과 공인ip 넣고 root 계정 넣고
맞는지 Test 후 success 나오면 됨)



Jenkins 서버에서 스프링부트 서버를 제어하여 스프링부트 컨테이너 실행하기
스프링부트 서버에서 docker pull 및 run

Dashboard

myapp Freestyle 아이템 선택

구성 탭 선택

빌드 환경

Send files or execute commands over SSH after the build runs 선택

SSH Server Name: Publish over SSH에 등록한 서버를 선택

Transfers

Exec command

```
docker login -u '도커허브아이디' -p '도커허브비번' docker.io
docker pull [dockerHub UserName]/[dockerHub Repository]:[version]
docker ps -q --filter name=[containerName] | grep -q . && docker rm -f $(docker ps -aq --filter name=[containerName])
docker run -d --name [containerName] -p 80:스프링부트포트번호 [dockerHub UserName]/[dockerHub Repository]:[version]
```

내꺼로 각자 수정

```
docker login -u 'hub아이디' -p 'password' docker.io
docker pull 도커허브아이디/react-docker:2.0
docker ps -q --filter name=docker-springboot | grep -q . && docker rm -f $(docker ps -aq --filter name=docker-springboot)
docker run -d --name docker-springboot -p 80:9002 도커허브아이디/react-docker:2.0
```