Naver Cloud
Docker + Jenkins 자동배포
구현 매뉴얼

2023.06.27.

- 비트 마켓 -

변경내역

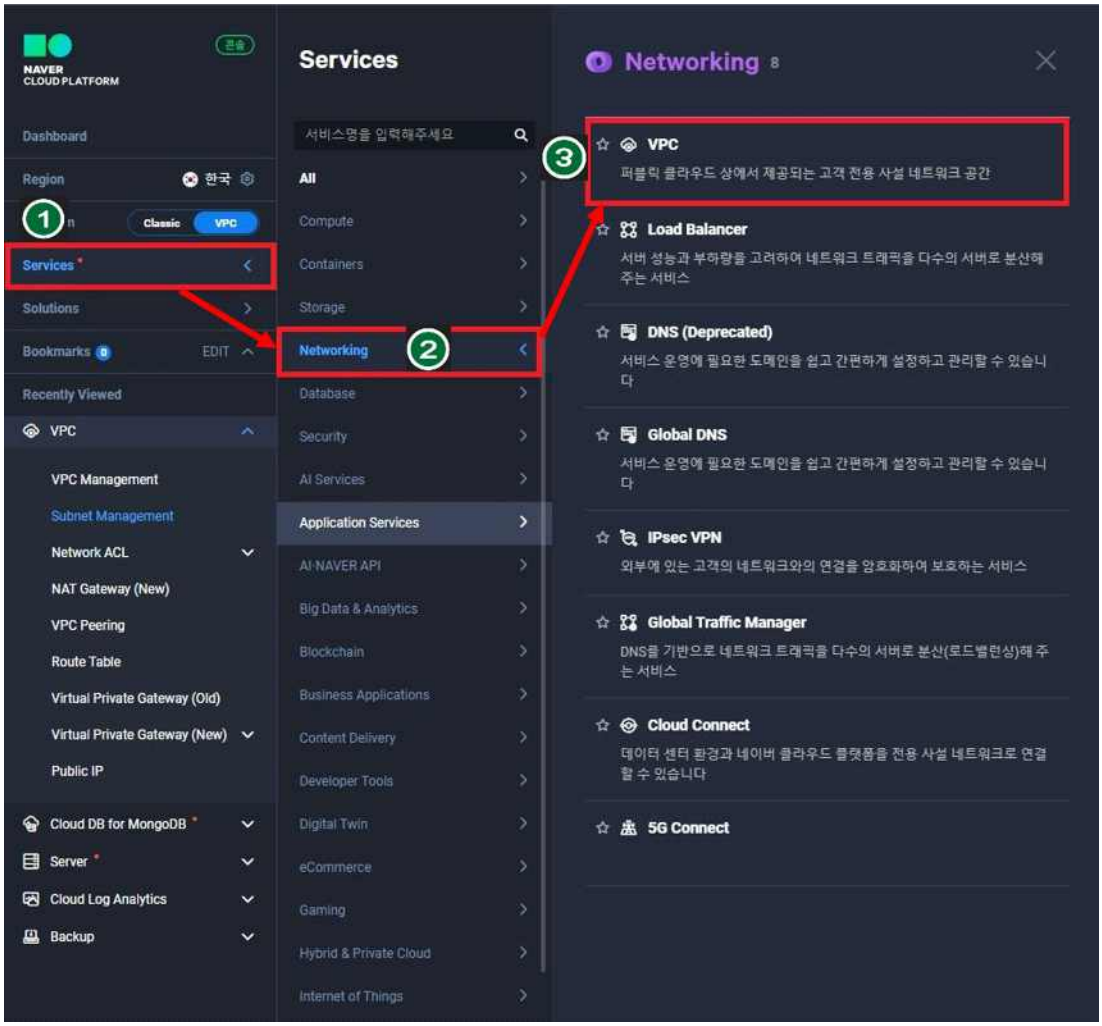
날짜	구분 (추가 / 수정 / 변경)	버전	내용
2023.06.27	추가	v1	초안 작성
2023.06.29	추가	v2	P34 - Docker에 Jenkins 권한 내용 추가

목 차

1 VPC 생성

※ 서버를 생성하기 이전, VPC와 Subnet 생성을 먼저 해야 한다.

○ 콘솔 창에서 Service → Networking → VPC 클릭



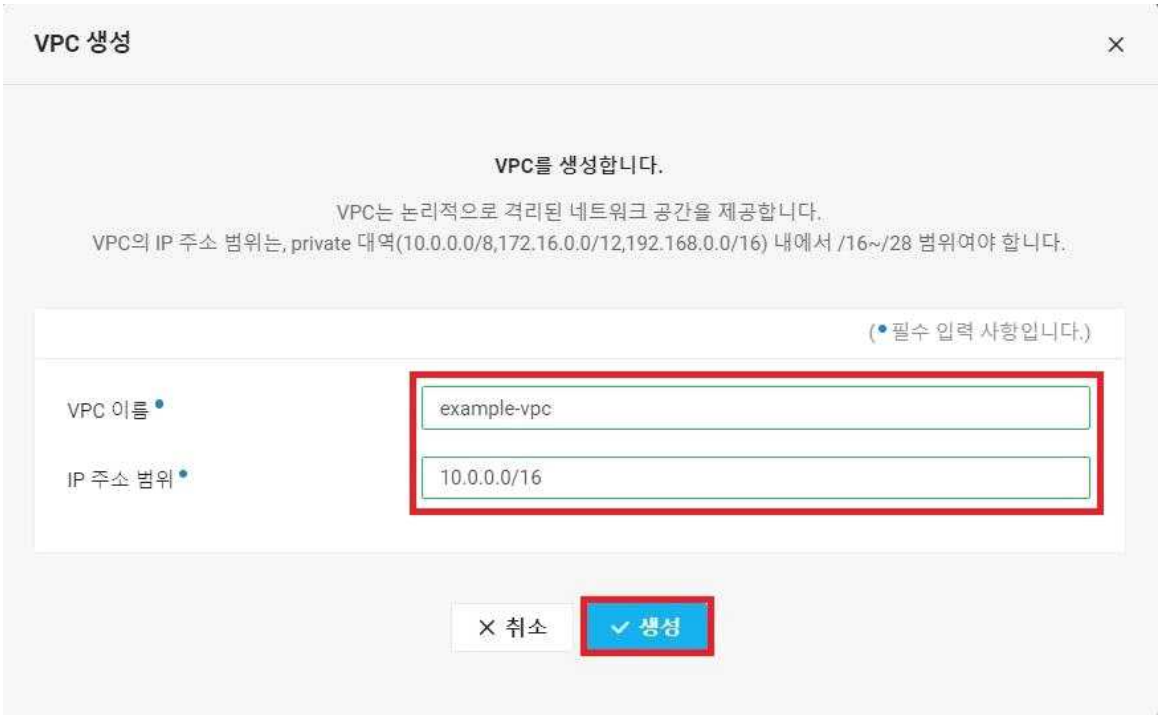
○ VPC 생성 버튼 클릭

○ VPC 생성창에서 VPC 이름과 IP 주소 범위를 입력하고 생성 버튼 클릭

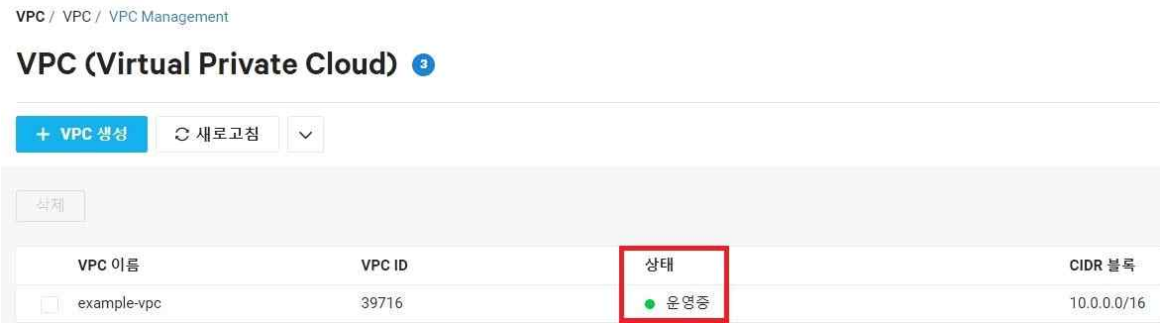
– 각각의 항목들은 다음과 같이 예시로 생성하였음

● VPC 이름: example-vpc

● IP 주소 범위: 10.0.0.0/16

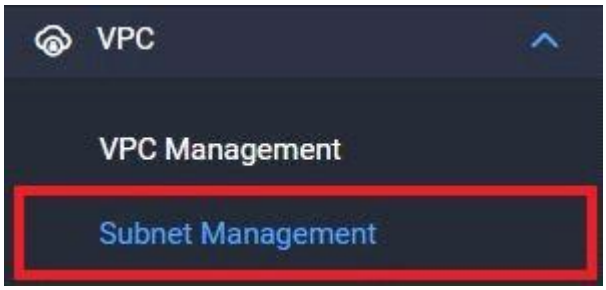


○ 생성 후 상태가 운영중 인지 확인



2 Subnet 생성

○ 왼쪽 메뉴의 VPC에서 Subnet Management 클릭

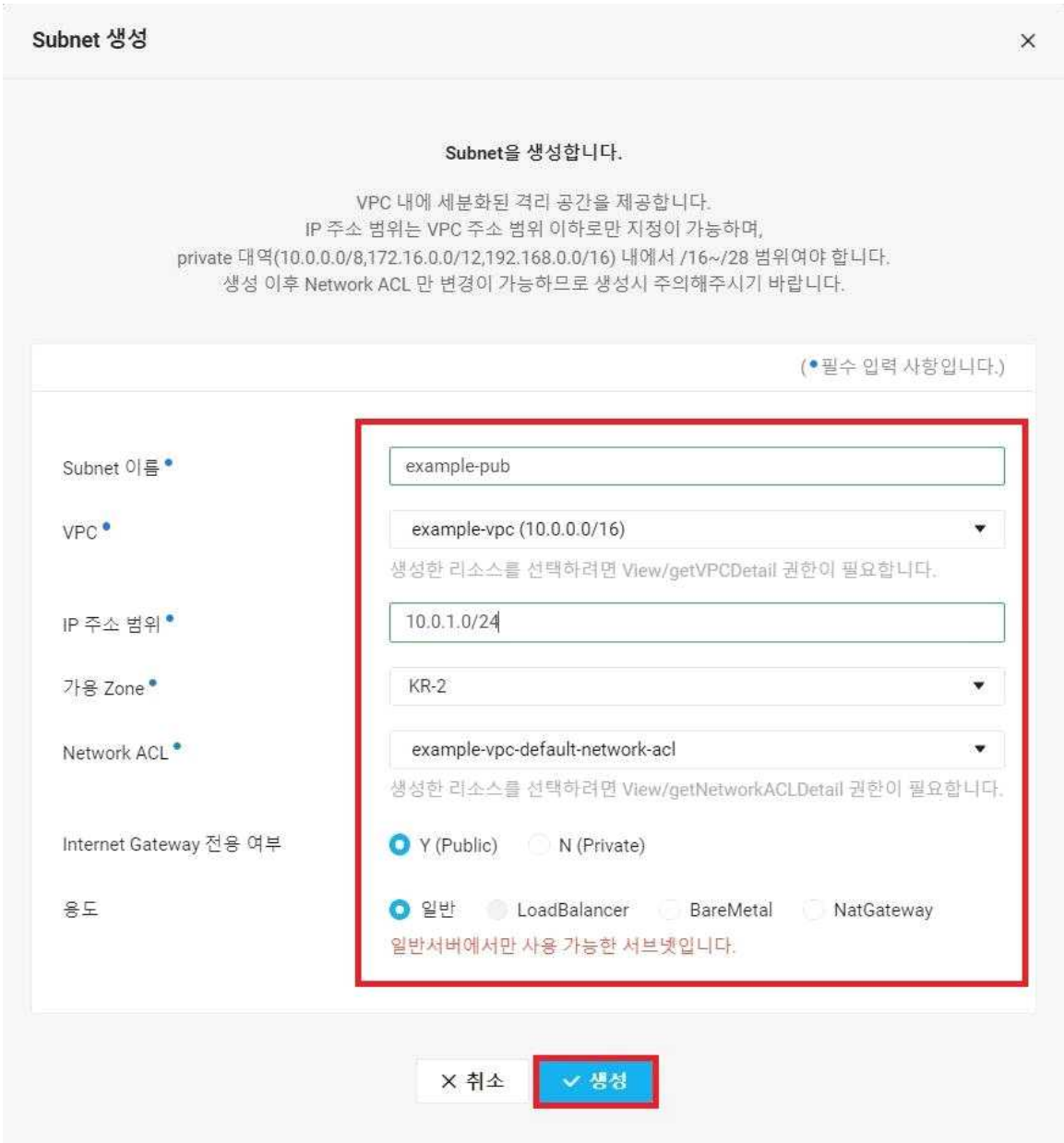


○ Subnet 생성 버튼 클릭

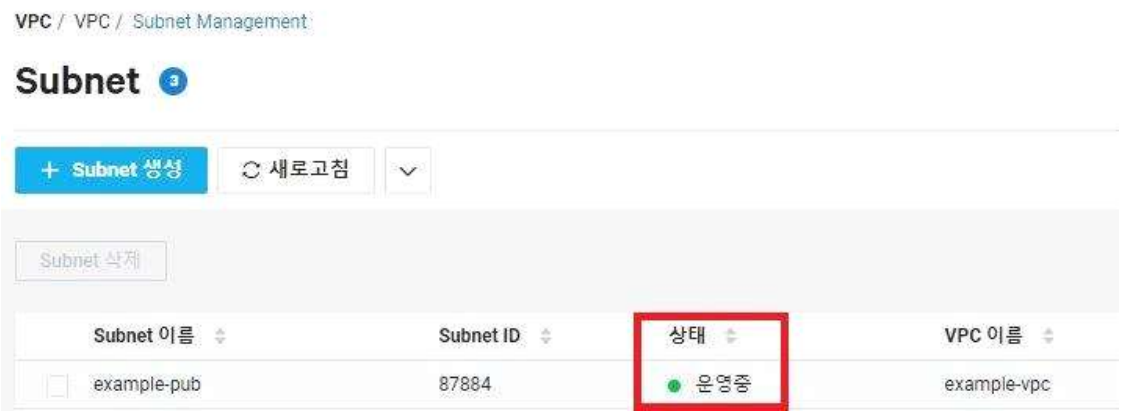
○ Subnet 생성창에서 각각의 항목들을 입력

– 각각의 항목들은 다음과 같이 예시로 생성하였음

- Subnet 이름: example-pub
- VPC: 앞에서 생성한 VPC 선택(example-vpc)
- IP 주소 범위: 10.0.1.0/24
- 가용 Zone: KR-2
- Network ACL: example-vpc-default-network-acl
- Internet Gateway 전용 여부: Y(Public)
- 용도: 일반

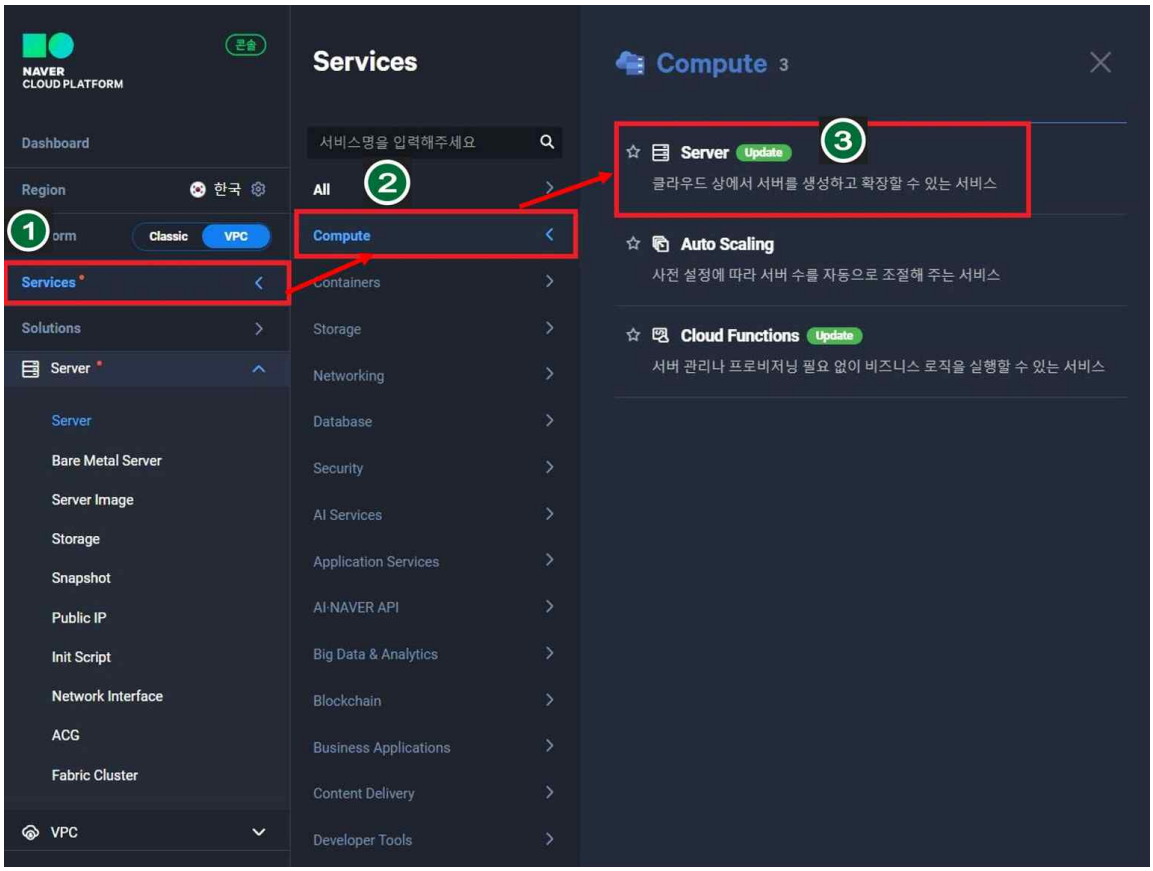


○ 생성 후 상태가 운영중 인지 확인



3 App Server 생성

○ 콘솔 창에서 Services → Compute → Server 클릭

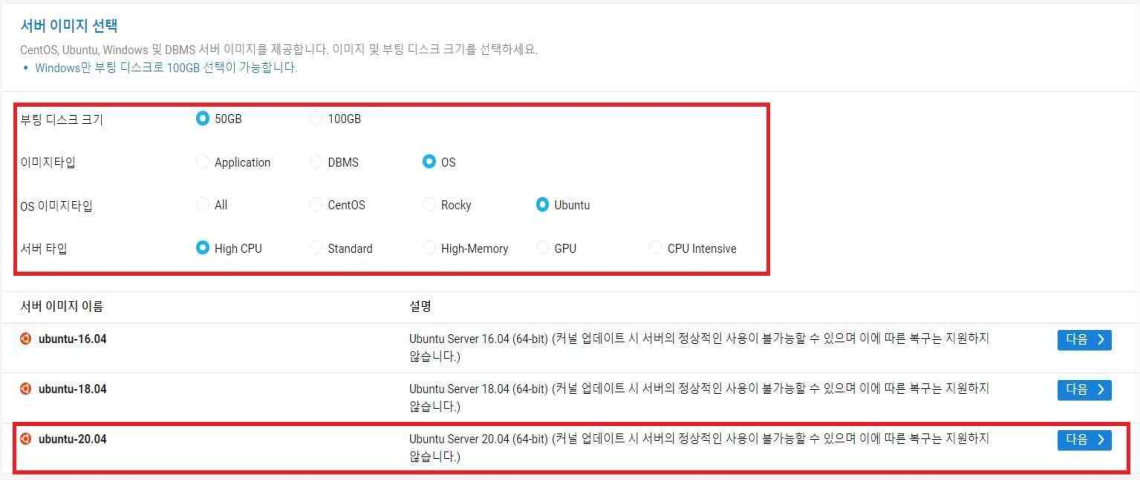


○ Server 생성 버튼 클릭

○ 서버 생성 단계에서 다음과 같이 설정하고 다음 버튼 클릭

– 각각의 항목들은 다음과 같이 예시로 설정하였음

- 부팅 디스크 크기: 50GB
- 이미지타입: OS
- OS 이미지타입: Ubuntu(우분투)
- 서버타입: High CPU
- 서버 이미지 이름: ubuntu-20.04



○ 서버 설정 단계에서 다음과 같이 설정하고 다음 버튼 클릭

– 각각의 항목들은 다음과 같이 예시로 생성하였음

- VPC: 앞에서 생성한 VPC 선택(example-vpc)
- Subnet: 앞에서 생성한 Subnet 선택(example-pub)
- 스토리지 종류: SSD
- 서버 세대: g2
- 서버 타입: [High CPU] vCPU 2개, 메모리 4GB, [SSD] 디스크 50GB [g2]
- 요금제 선택: 시간 요금제
- 서버 개수: 1
- 서버 이름: example-server
- Network Interface: 자동할당
- 공인 IP: 새로운 공인 IP 할당
- 물리 배치 그룹: 미사용
- 반납 보호: 해제
- 메모: 없음
- Script 선택: 선택없음

○ 인증키 설정에서 선택 후 다음 버튼 클릭

– 보유하기 있는 인증키 이용 선택 시 인증키 선택 후 다음 버튼 클릭

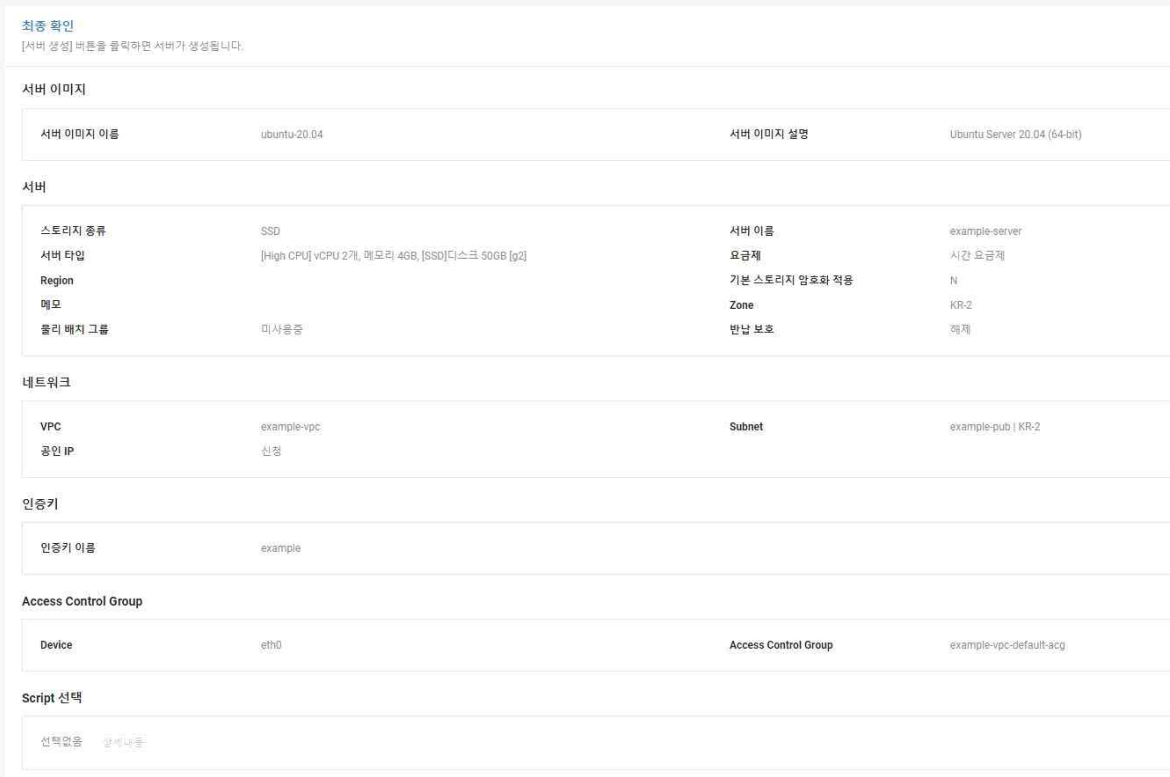
– 새로운 인증키 생성 선택 시 인증키 이름을 입력 후 인증키 생성 및 저장 한 후 다음 버튼 클릭

○ 네트워크 접근 설정 단계에서 eth0 선택 후 다음 버튼 클릭

– eth0은 기본 값을 선택하였음

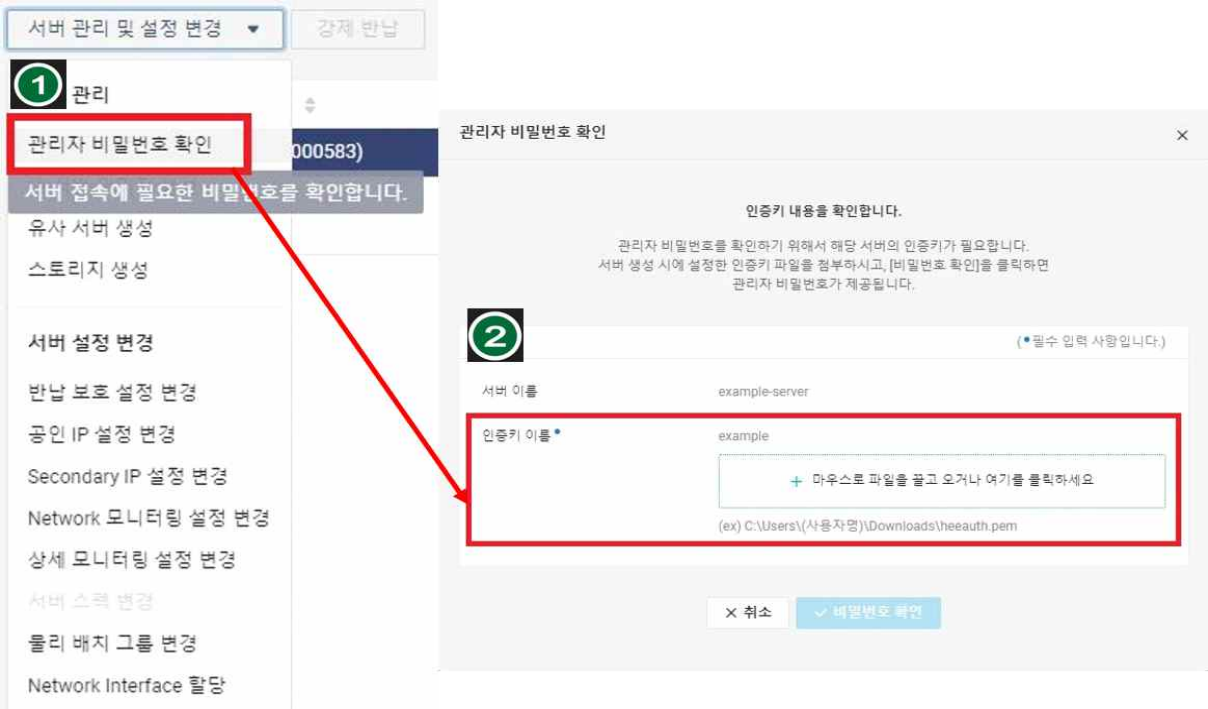
○ 최종 단계에서 설정한 값들이 맞는지 확인 후 생성 버튼 클릭

– 생성 시간은 최장 약 30분 정도 소요

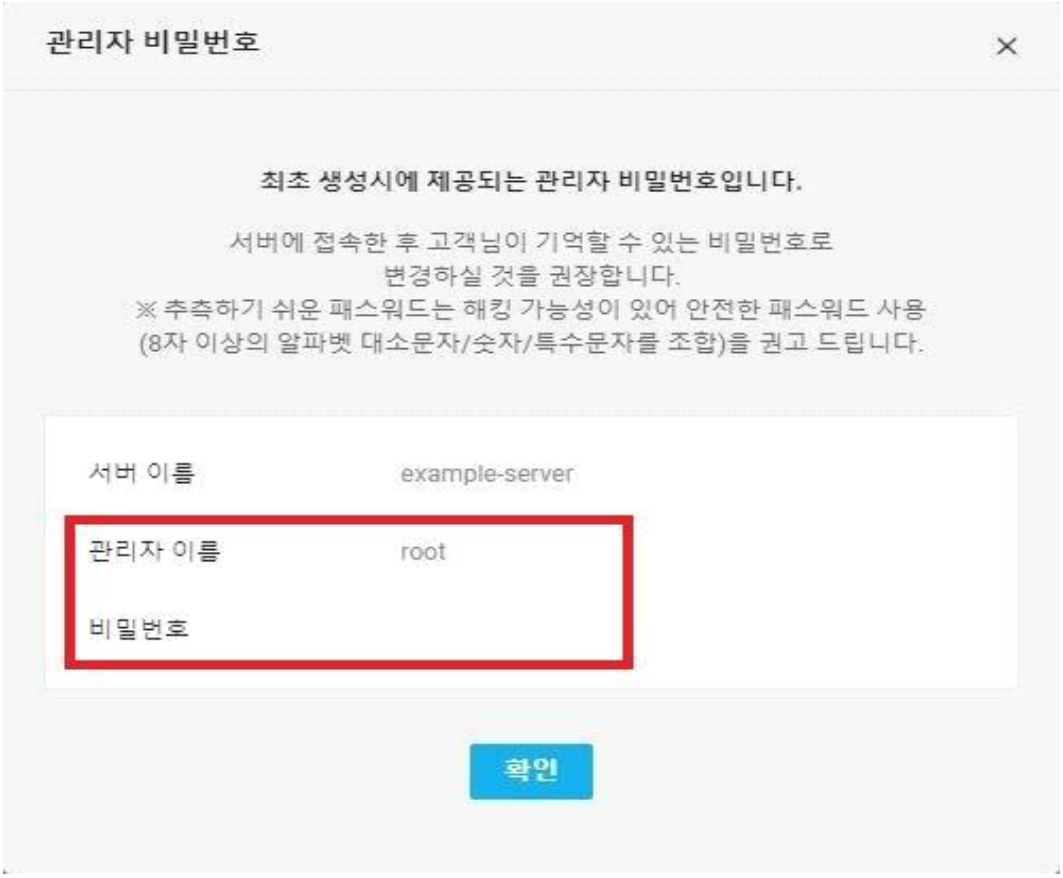


4 Server 설정

○ 콘솔에서 Server로 돌아와서 생성이 되었는지 확인 후 서버에 접속하기 위해 관리자 비밀번호 확인 → 인증키 내용 확인 진행

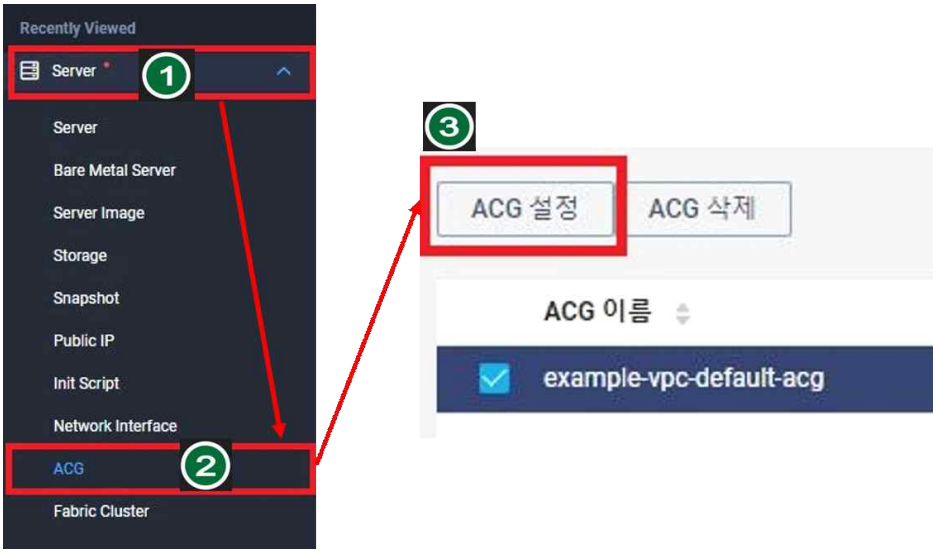


○ 관리자 이름과 비밀번호를 확인



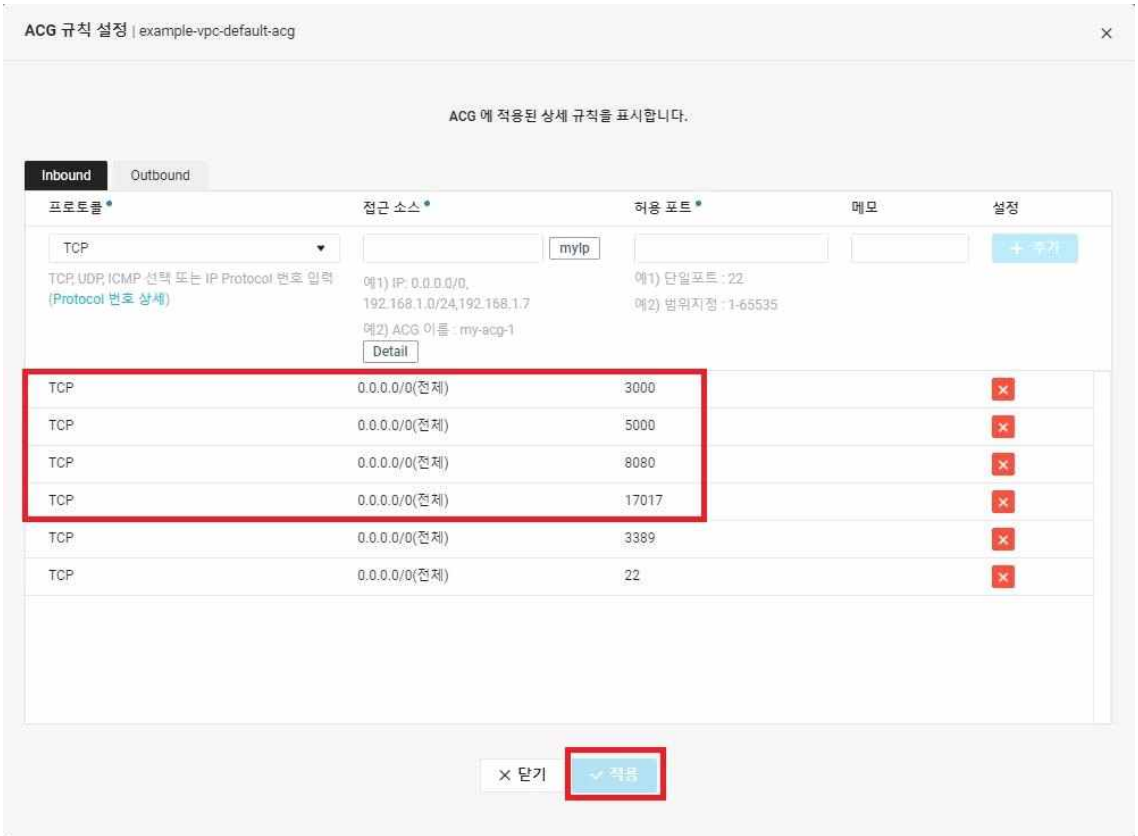
○ 콘솔에서 Server → ACG → ACG 설정 클릭

- ACG 이름은 예시로 example-vpc-default-acg



○ ACG 규칙 설정에서 다음과 같이 설정하고 적용 버튼 클릭

- TCP 0.0.0.0/0 3000: React
- TCP 0.0.0.0/0 5000: Node.js Server
- TCP 0.0.0.0/0 8080: Jenkins
- TCP 0.0.0.0/0 17017: MongoDB



○ Server에서 공인 IP를 확인 후 접속 진행

– 여기서는 Visual Studio Code의 remote - SSH 확장으로 접속했음

○ root 관리자 계정은 보안상 취약하므로 계정을 생성 후 진행

– 여기서는 계정을 student로 생성하였음

● adduser: 사용자 추가

```
root@example-server:~# adduser student
Adding user `student' ...
Adding new group `student' (1002) ...
Adding new user `student' (1002) with group `student' ...
Creating home directory `/home/student' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for student
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@example-server:~#
```

○ 계정 생성 후 sudo 명령어를 사용하기 위해 다음과 같이 설정

● visudo: root 권한을 획득하기 위해 리눅스 시스템에서 /etc/sudoers 파일을 안전하게 편집하기 위한 명령어

● root ALL=(ALL:ALL) ALL 아래에

[계정명] ALL=(ALL:ALL) ALL 추가

```
GNU nano 4.8                                                    /etc/sudoers.tmp

# User privilege specification
root    ALL=(ALL:ALL) ALL
student   ALL=(ALL:ALL) ALL

# 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

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

○ 설정이 완료되었으면 다음과 같이 student 계정으로 사용자 전환

● su – [계정명]: [계정명]으로 사용자 전환

```
문제    출력    디버그 콘솔    터미널    포트

root@example-server:~# su - student
student@example-server:~$
```

5 Docker 설치

※ Docker는 개발 시 application을 쉽고 빠르게 구축, 공유 및 실행할 수 있는 소프트웨어이다.

○ 필수 패키지 설치를 위해 다음과 같이 입력

– 맨 처음 sudo 명령어를 사용할 때 비밀번호를 입력

● sudo apt-get update

● sudo apt-get install -y \
ca-certificates \
curl \
gnupg \
lsb-release

```
student@example-server:~$ sudo apt-get update
[sudo] password for student:
Hit:1 http://repo.ncloud.com/ubuntu focal InRelease
Hit:2 http://repo.ncloud.com/ubuntu focal-updates InRelease
Hit:3 http://repo.ncloud.com/ubuntu focal-backports InRelease
Hit:4 http://repo.ncloud.com/ubuntu focal-security InRelease
Reading package lists... Done
student@example-server:~$ sudo apt-get install -y \
>     ca-certificates \
>     curl \
>     gnupg \
>     lsb-release
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu2).
lsb-release set to manually installed.
The following additional packages will be installed:
  dirmngr gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv libcurl4
```

○ Docker의 공식 GPG키를 추가하고 공식 APT 저장소도 추가

● curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

● echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
student@example-server:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
File '/usr/share/keyrings/docker-archive-keyring.gpg' exists. Overwrite? (y/N) y
student@example-server:~$ echo \
>   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
>   $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

○ 시스템 패키지를 업데이트하고 Docker를 설치

- sudo apt-get update
- sudo apt-get install -y docker-ce docker-ce-cli containerd.io

```
student@example-server:~$ sudo apt-get update
Hit:1 http://repo.ncloud.com/ubuntu focal InRelease
Hit:2 http://repo.ncloud.com/ubuntu focal-updates InRelease
Hit:3 http://repo.ncloud.com/ubuntu focal-backports InRelease
Hit:4 http://repo.ncloud.com/ubuntu focal-security InRelease
Get:5 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [30.3 kB]
Fetched 88.0 kB in 0s (194 kB/s)
Reading package lists... Done
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en_US) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en_US) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
student@example-server:~$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin pigz slirp4netns
0 upgraded, 8 newly installed, 0 to remove and 220 not upgraded.
Need to get 111 MB of archives.
After this operation, 401 MB of additional disk space will be used.
```

○ 설치 후, Docker 설치가 제대로 되어있는지 명령어로 확인

※ Docker version

```
student@example-server:~$ docker version
Client: Docker Engine - Community
 Version:           24.0.2
 API version:       1.43
 Go version:        go1.20.4
 Git commit:        cb74dfc
 Built:             Thu May 25 21:52:22 2023
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the
run/docker.sock: connect: permission denied
```

○ permission denied 문제가 발생 시, 해결을 위해 계정을 사용자 그룹에 추가

※ sudo groupadd docker

※ sudo usermod -aG docker $USER

※ newgrp docker

```
student@example-server:~$ sudo groupadd docker
groupadd: group 'docker' already exists
student@example-server:~$ sudo usermod -aG docker $USER
student@example-server:~$ newgrp docker
student@example-server:~$
```

○ Docker를 활성화

- sudo systemctl enable docker
- sudo systemctl start docker
- sudo systemctl enable containerd
- sudo systemctl start containerd

```
student@example-server:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
student@example-server:~$ sudo systemctl start docker
student@example-server:~$ sudo systemctl enable containerd
student@example-server:~$ sudo systemctl start containerd
student@example-server:~$
```

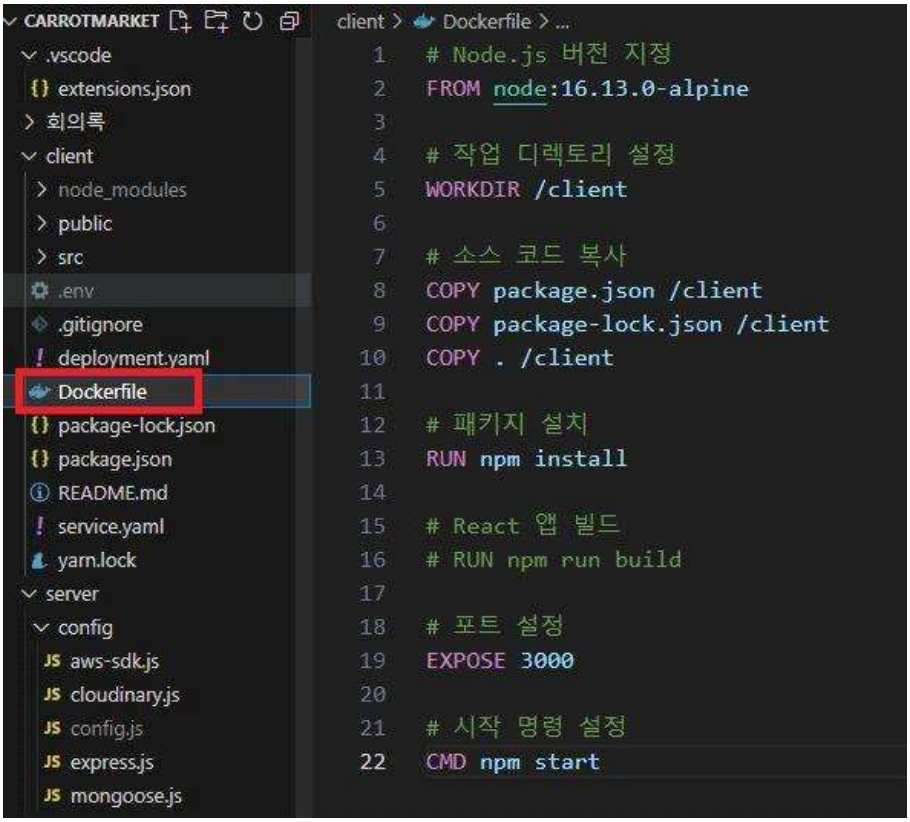
○ hello-world 컨테이너를 실행해서 정상적으로 설치가 되었는지 확인

- sudo docker run --rm hello-world

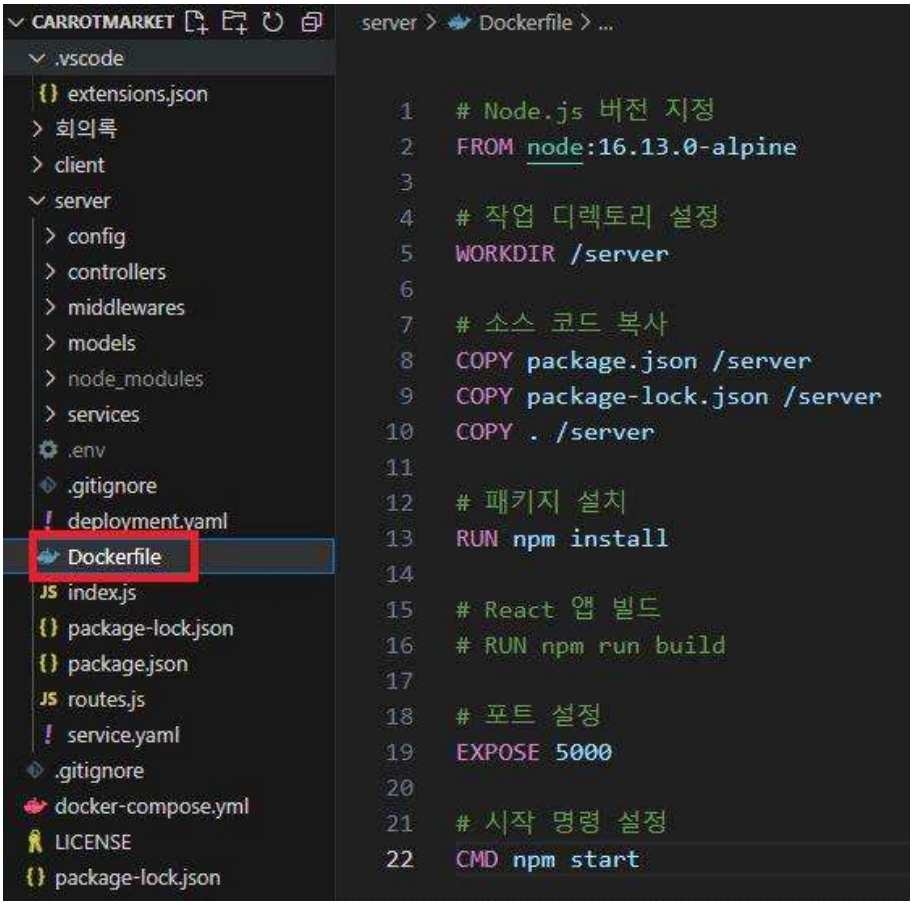
```
student@example-server:~$ sudo docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:a13ec89cdf897b3e551bd9f89d499db6ff3a7f44c5b9eb8bca40da20eb4ea1fa
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

○ 프로젝트 내 client · server 폴더 내부에 다음과 같이 Dockerfile 작성

● client 내부 Dockerfile



● server 내부 Dockerfile



6 Jenkins 설치

※ Jenkins(젠킨스)는 지속적인 통합(Continuous Integration) 및 지속적인 제공(Continuous Delivery)을 지원하는 오픈 소스 자동화 도구이다. 소프트웨어 개발 프로세스에서 품질 향상과 효율성을 도모하기 위해 사용된다.

○ Jenkins 설치에 앞서 Java 설치

- Jenkins는 Java 응용 프로그램이며 시스템에 Java 8 이상을 설치해야 한다.
- 여기서는 Java Platform의 오픈 소스 구현인 OpenJDK 11을 설치했다.

● sudo apt-get update

● sudo apt-get install openjdk-11-jdk

```
student@example-server:~$ sudo apt-get update
Hit:1 http://repo.ncloud.com/ubuntu focal InRelease
Hit:2 http://repo.ncloud.com/ubuntu focal-updates InRelease
Hit:3 http://repo.ncloud.com/ubuntu focal-backports InRelease
Hit:4 http://repo.ncloud.com/ubuntu focal-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Reading package lists... Done
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en_US) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en_US) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:62 and /etc/apt/sources.list.d/docker.list:1
student@example-server:~$ sudo apt-get install openjdk-11-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

○ Java 설치가 잘 되어있는지 확인

● java -version

```
student@example-server:~$ java -version
openjdk version "11.0.19" 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu120.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu120.04.1, mixed mode, sharing)
student@example-server:~$
```

○ Jenkins 설치를 위해 Repository key를 추가하고, 서버의 sources.list에 Jenkins 패키지 저장소를 추가

※ 2023년 3월 28일부터 Linux 설치 패키지에 대한 새로운 레포지토리 서명 키를 사용하므로, Jenkins를 설치하기 전에 새로운 서명 키로 설치해야 한다.

● curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null

● echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

```
student@example-server:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
>   /usr/share/keyrings/jenkins-keyring.asc > /dev/null
student@example-server:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
>   https://pkg.jenkins.io/debian binary/ | sudo tee \
>   /etc/apt/sources.list.d/jenkins.list > /dev/null
student@example-server:~$
```

○ 시스템 패키지를 업데이트하고 Jenkins를 설치

- sudo apt update
- sudo apt install jenkins

```
student@example-server:~$ sudo apt update
Hit:1 http://repo.ncloud.com/ubuntu focal InRelease
Hit:2 http://repo.ncloud.com/ubuntu focal-updates InRelease
Hit:3 http://repo.ncloud.com/ubuntu focal-backports InRelease
Hit:4 http://repo.ncloud.com/ubuntu focal-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Ign:6 https://pkg.jenkins.io/debian binary/ InRelease
Get:7 https://pkg.jenkins.io/debian binary/ Release [2,044 B]
Get:8 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Get:9 https://pkg.jenkins.io/debian binary/ Packages [54.6 kB]
Fetched 57.5 kB in 3s (22.2 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
220 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
student@example-server:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 220 not upgraded.
Need to get 88.8 MB of archives.
After this operation, 89.3 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian binary/ jenkins 2.412 [88.8 MB]
Fetched 88.8 MB in 6s (15.2 MB/s)
Selecting previously unselected package jenkins.
dpkg: warning: files list file for package 'sudo' missing; assuming package has no files currently installed
(Reading database ... 79389 files and directories currently installed.)
Preparing to unpack .../archives/jenkins_2.412_all.deb ...
Unpacking jenkins (2.412) ...
Setting up jenkins (2.412) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for systemd (245.4-4ubuntu3.15) ...
student@example-server:~$
```

○ systemctl 명령을 사용하여 Jenkins 서버를 시작, 활성화, 상태를 확인

– active 상태 체크

- sudo systemctl start jenkins
- sudo systemctl enable jenkins
- sudo systemctl status jenkins

```
student@example-server:~$ sudo systemctl start jenkins
student@example-server:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable jenkins
student@example-server:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-29 10:10:06 KST; 17min ago
   Main PID: 671 (java)
      Tasks: 42 (limit: 4590)
     Memory: 1.1G
     CGroup: /system.slice/jenkins.service
             └─671 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

○ Jenkins 서버의 포트 번호 변경은 다음과 같이 진행

- port for HTTP connector에서 변경
- 여기서는 default의 값인 8080으로 진행

● sudo vi /etc/default/jenkins

```
student@example-server:~$ sudo vi /etc/default/jenkins
```

```
# UMASK=027

# port for HTTP connector (default 8080; disable with -1)
HTTP_PORT=8080


# servlet context, important if you want to use apache proxying
PREFIX=/$NAME

# arguments to pass to jenkins.
# full list available from java -jar jenkins.war --help
# --javaHome=$JAVA_HOME
# --httpListenAddress=$HTTP_HOST (default 0.0.0.0)
# --httpPort=$HTTP_PORT (default 8080; disable with -1)
# --httpsPort=$HTTP_PORT
# --argumentsRealm.passwd.$ADMIN_USER=[password]
# --argumentsRealm.roles.$ADMIN_USER=admin
# --webroot=~/.jenkins/war
# --prefix=$PREFIX

JENKINS_ARGS="--webroot=/var/cache/$NAME/war --httpPort=$HTTP_PORT"
```

○ 방화벽에서 Jenkins의 기본 포트 번호인 8080을 열어주고 상태 확인

● sudo ufw allow 8080

● sudo ufw status

```
student@example-server:~$ sudo ufw allow 8080
Skipping adding existing rule
Skipping adding existing rule (v6)
student@example-server:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
8080                       ALLOW       Anywhere
8080 (v6)                  ALLOW       Anywhere (v6)
```

○ Jenkins의 서비스를 재시작하고 상태 확인

- active 상태 체크

● sudo service jenkins restart

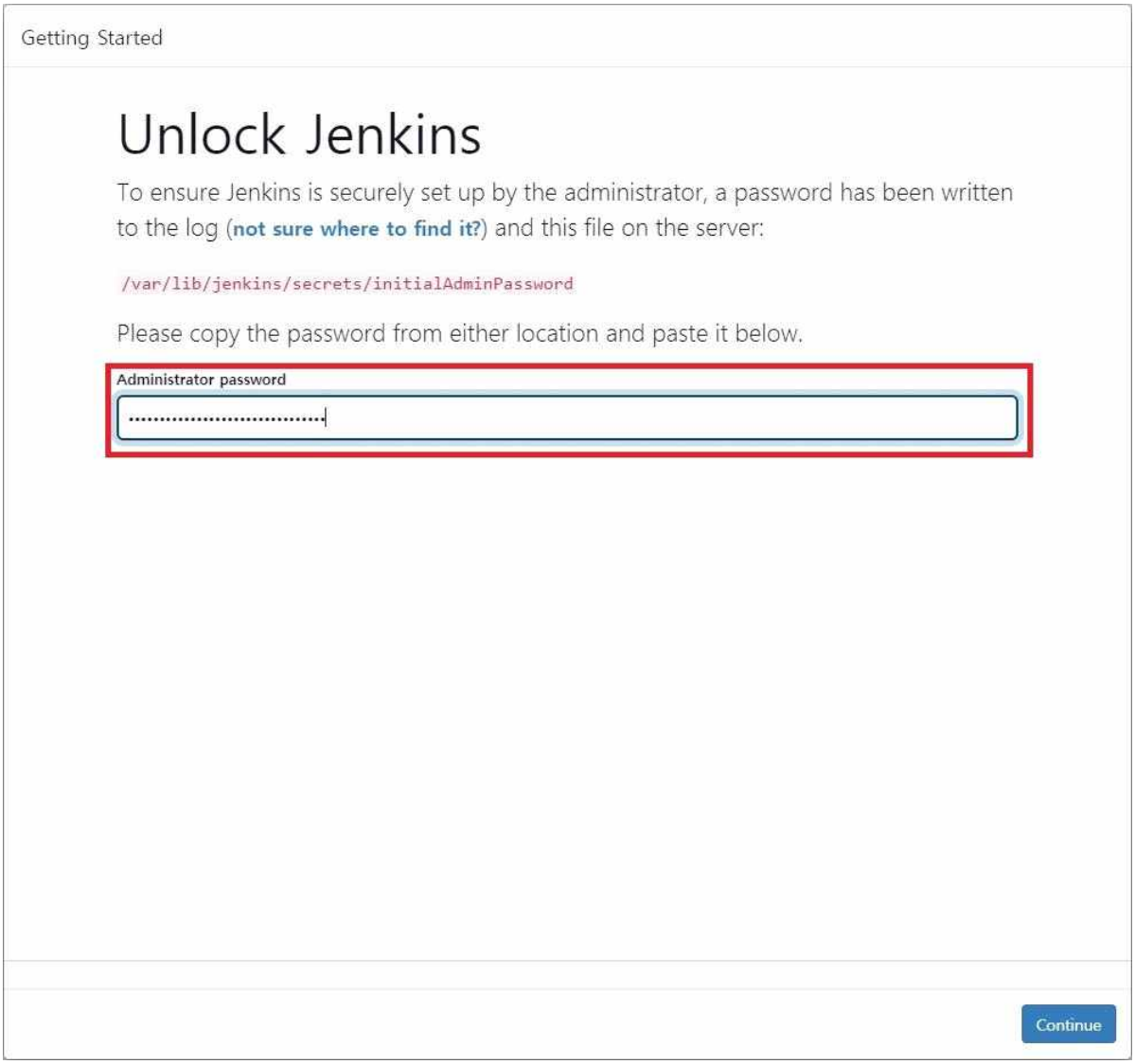
● sudo systemctl status jenkins

```
student@example-server:~$ sudo service jenkins restart
student@example-server:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-29 11:09:03 KST; 36s ago
   Main PID: 6527 (java)
      Tasks: 49 (limit: 4590)
     Memory: 1.1G
     CGroup: /system.slice/jenkins.service
             └─6527 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

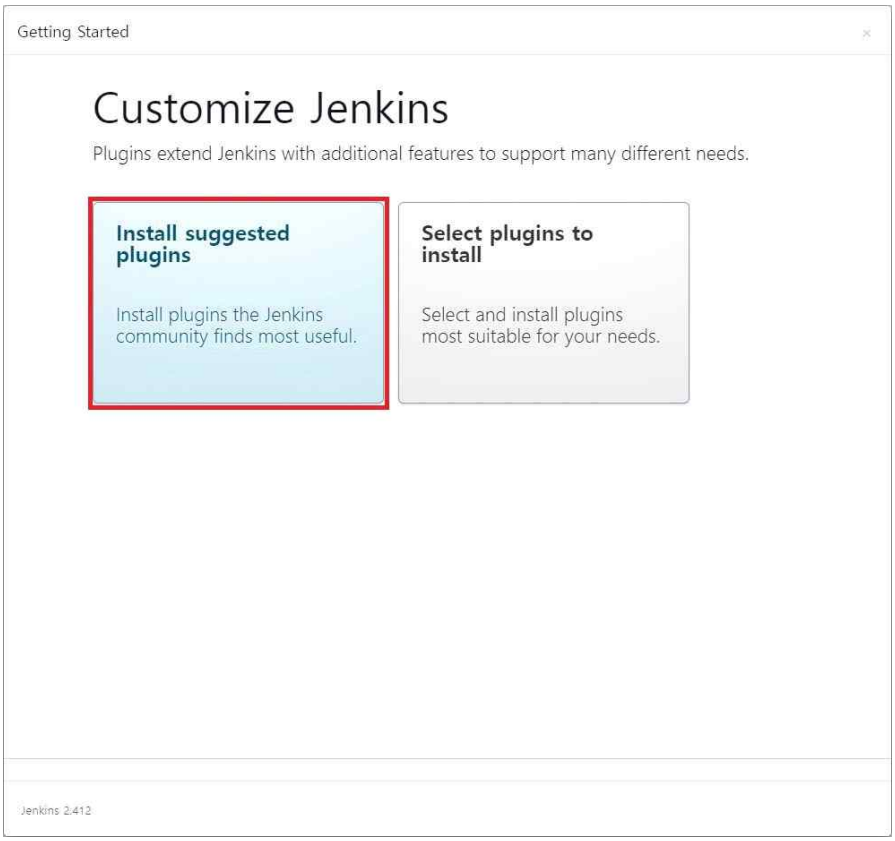
○ 주소창에 http://[서버URL]:[포트] URL을 입력해 Jenkins에 접속

- Administrator password는 sudo cat /var/lib/jenkins/secrets/initialAdminPassword 에서 확인

```
student@example-server:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
[sudo] password for student:
```

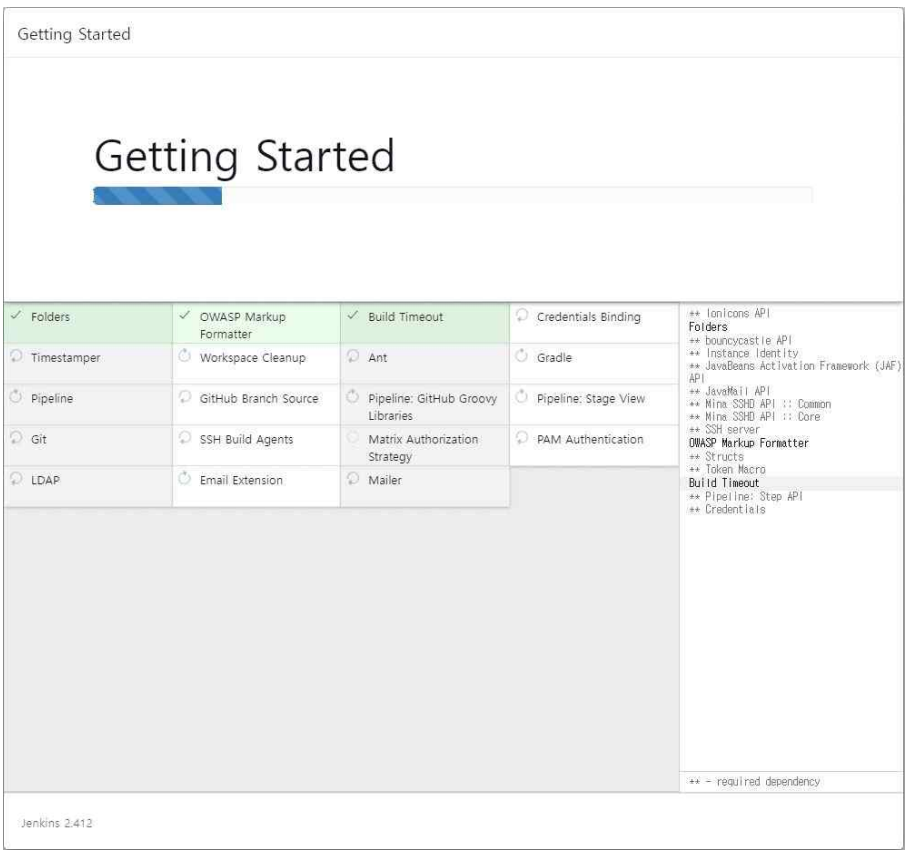


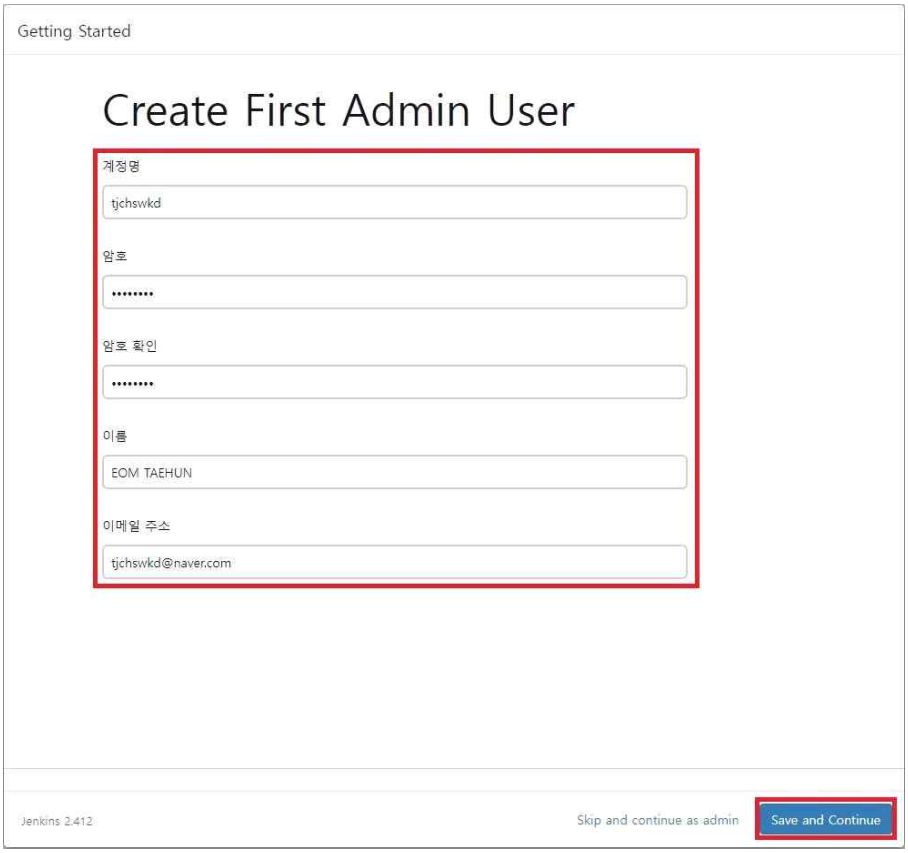
○ Customize Jenkins에서 Install suggested plugins 선택



○ 설치가 진행되고 끝나면 Create First Admin User에서 계정 설정 진행

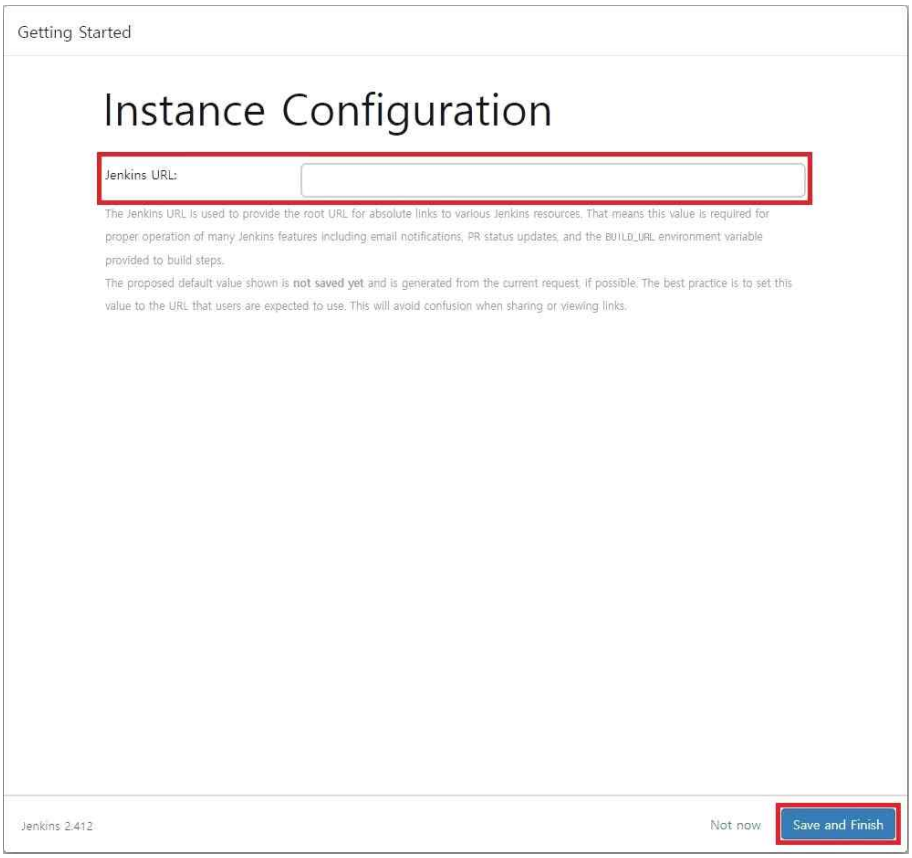
– 계정은 예시로 설정하였음



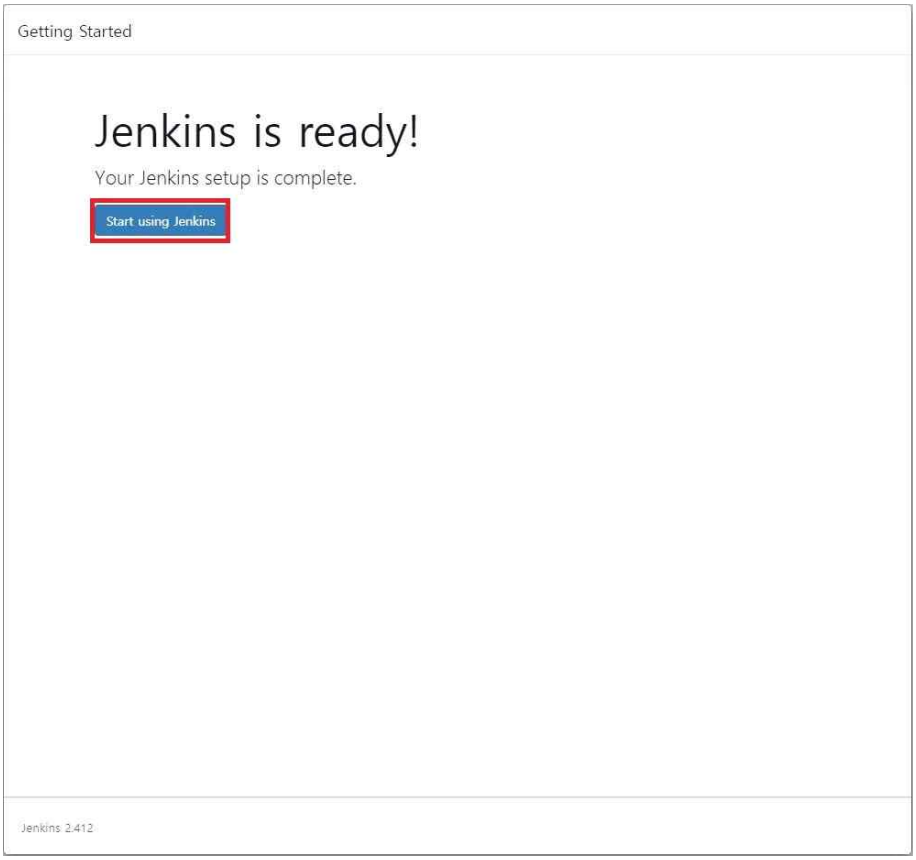


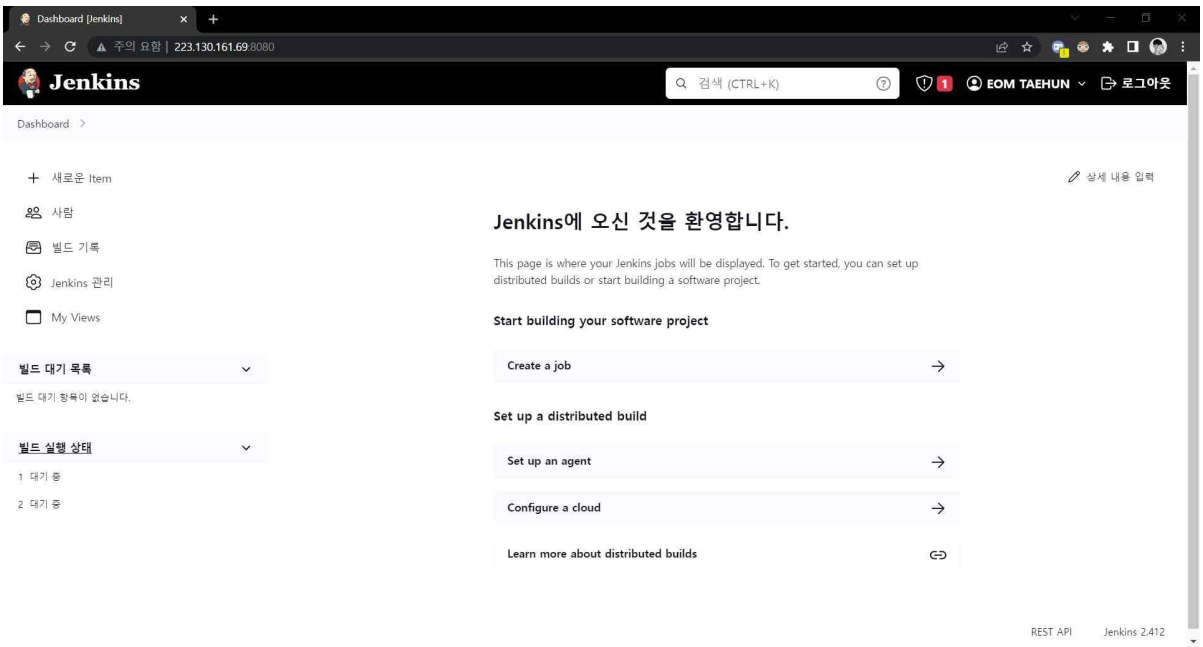
○ Instance Configuration에서 Jenkins URL 설정

– 여기서는 Jenkins URL을 default 값으로 설정하였음



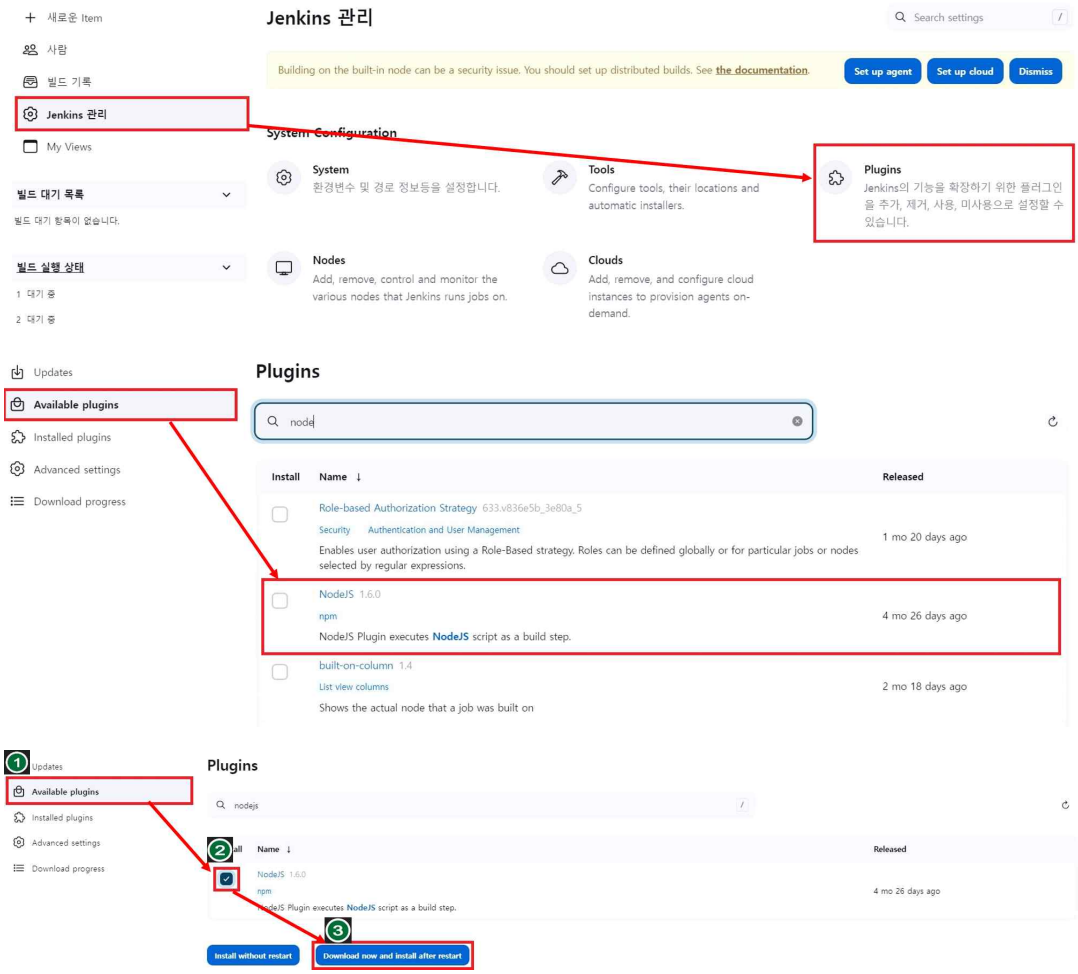
○ 모든 설정이 완료되면 Jenkins is ready!가 나오고 Start using Jenkins를 누르면 Jenkins 설치 완료





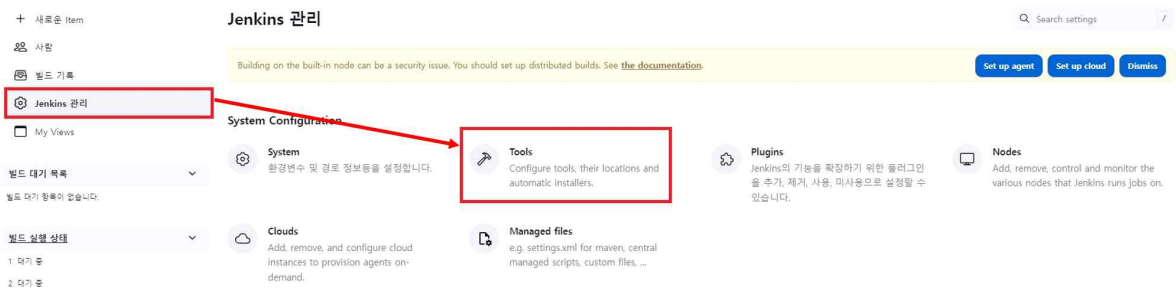
6 Jenkins Build 설정

○ Jenkins 관리 → Plugins → Available plugins → NodeJS 설치



○ Download progress에서 '설치가 끝나고 실행중인 작업이 없으면 Jenkins 재시작' 체크

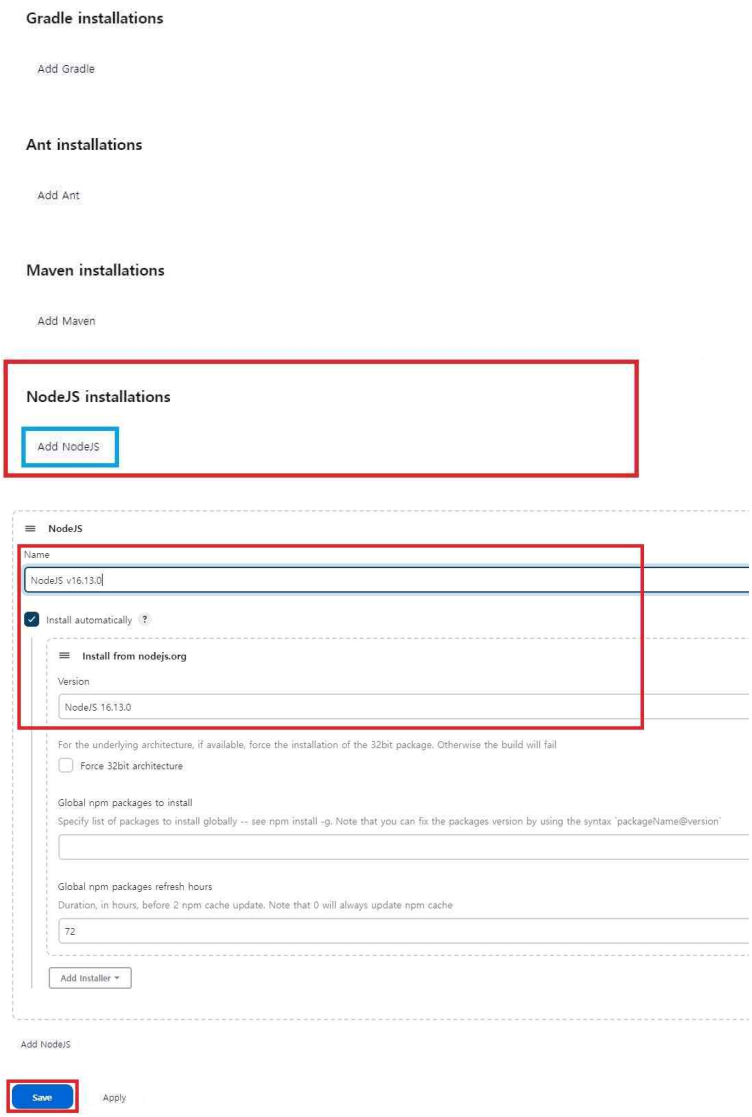
○ Jenkins 재로그인 → Jenkins 관리 → Tools 클릭



○ Tools 밑 부분에 NodeJS installations → Add NodeJS 클릭하고 다음과 같이 설정

– 각각의 항목들은 다음과 같이 설정하였음

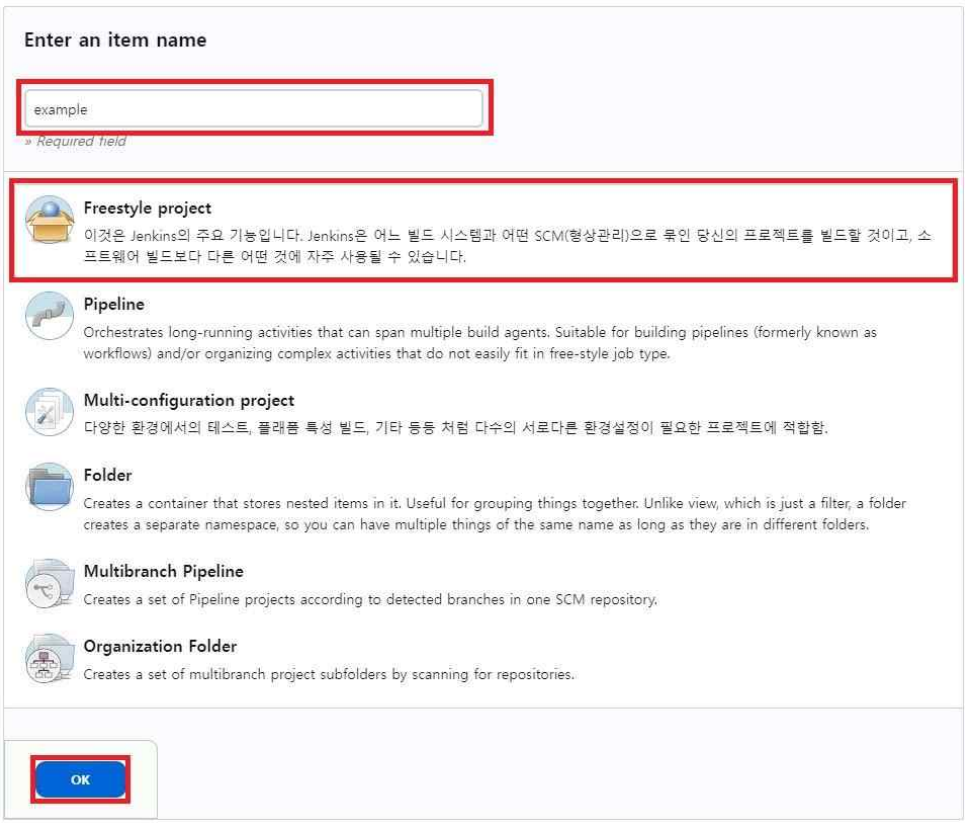
- Name: NodeJS v16.13.0
- Install automatically 체크
- Version: NodeJS 16.13.0



○ Jenkins 메인 화면 → Create a job 클릭

○ Enter an item name에서 이름을 입력하고 'Freestyle project'를 선택하고 OK 클릭

- item name은 예시로 example로 설정하였음

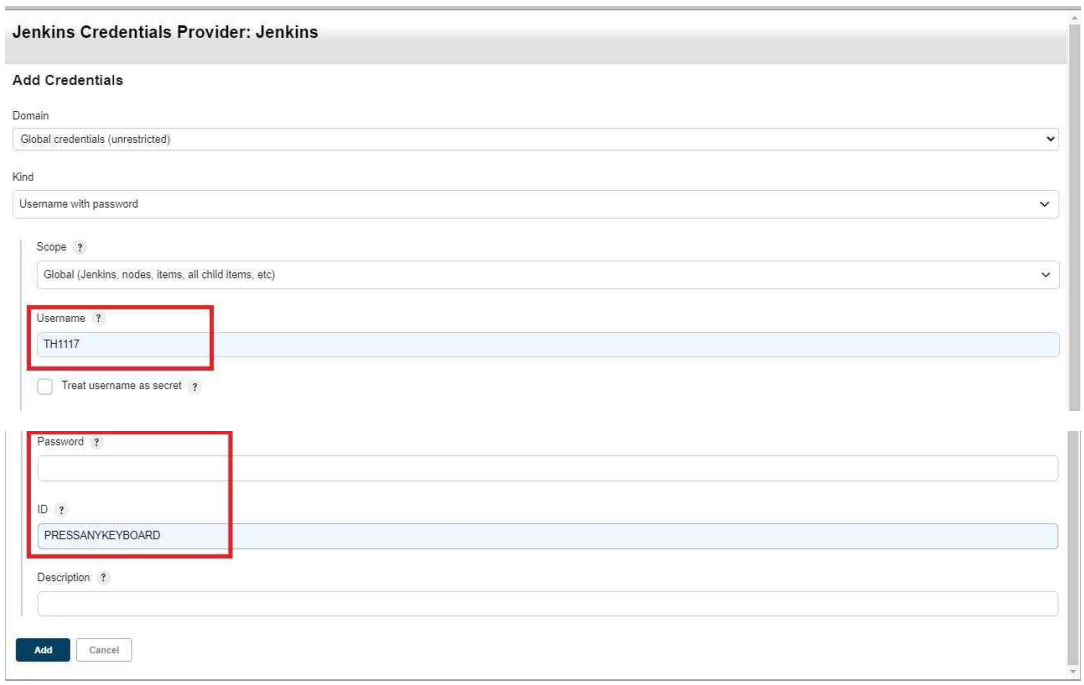


○ General에서 각각의 항목들에 대해 다음과 같이 설정

● GitHub project: 프로젝트의 GitHub 주소

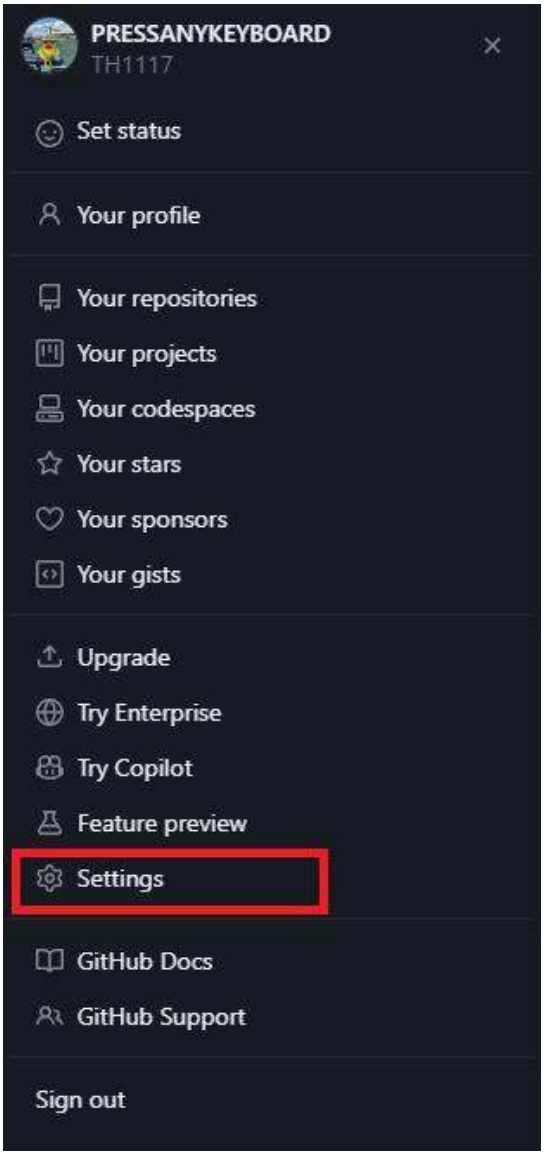
● 소스코드 관리 → Git → Add Jenkins

● Add Jenkins 설정

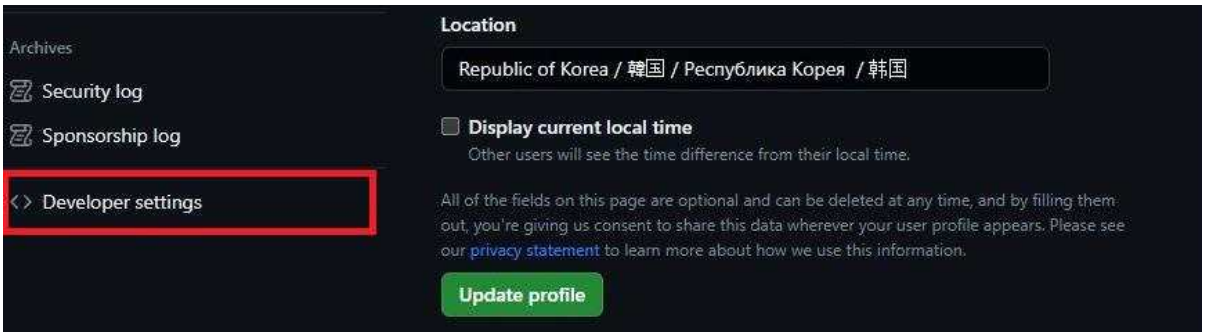


※ Add Jenkins의 Password 입력 방법

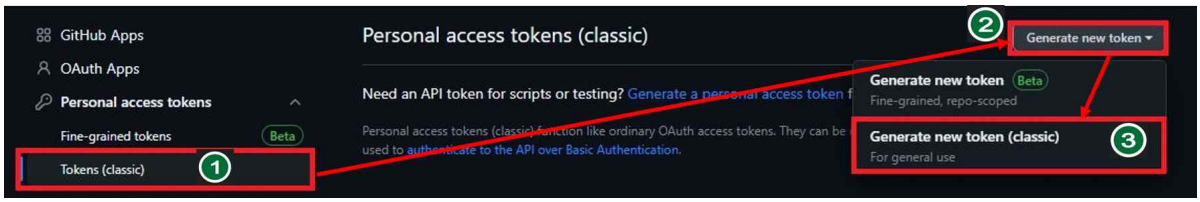
● 자기자신의 GitHub 프로필 → Settings 클릭



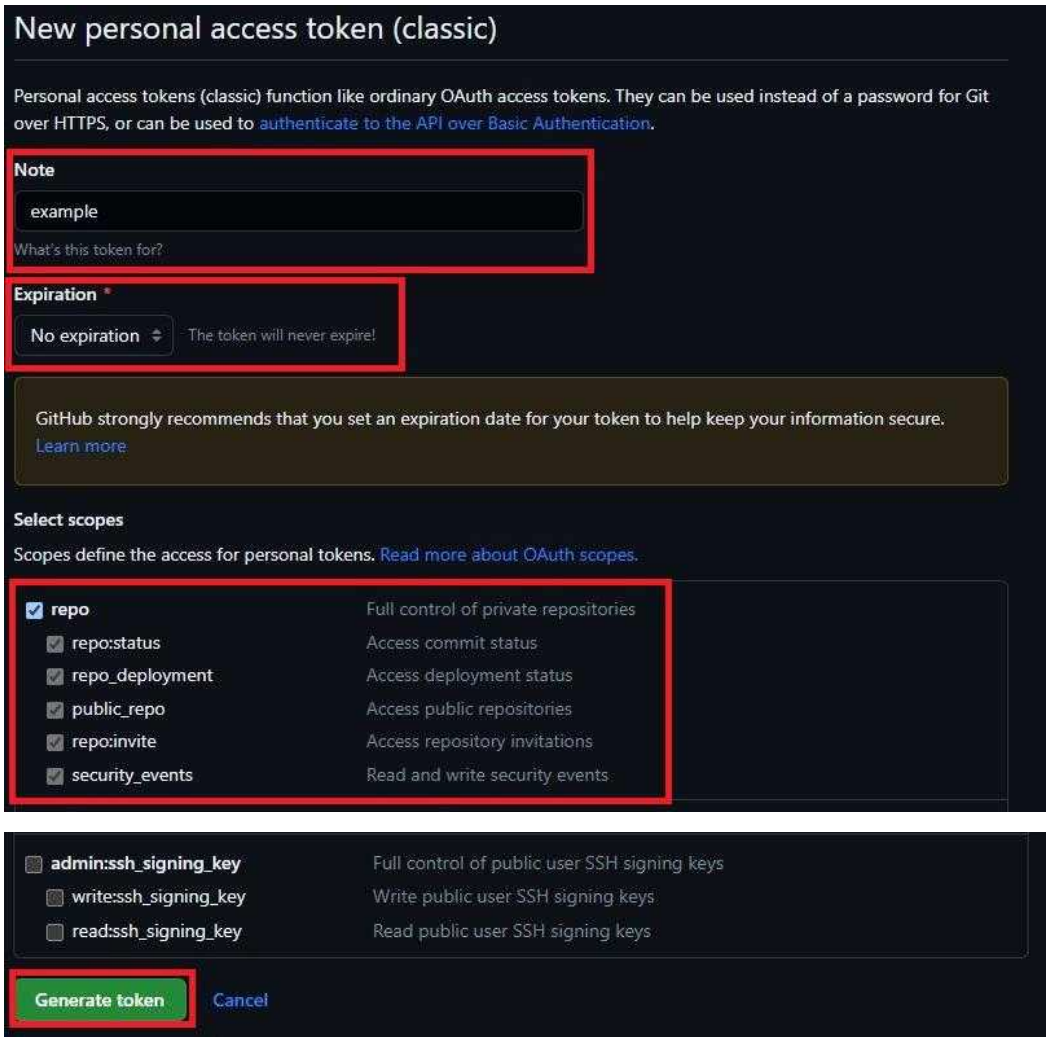
● 좌측 하단의 Developer settings 클릭



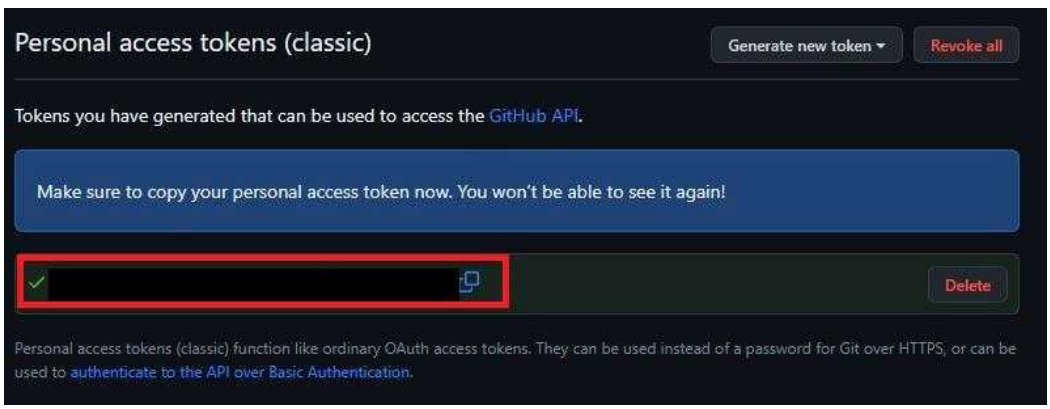
● Tokens(classic) → Generate new token → Generate new token(classic) 클릭



● Note, Expiration을 설정하고 Select scopes에서 repo만 체크하고 Gernerate token 클릭

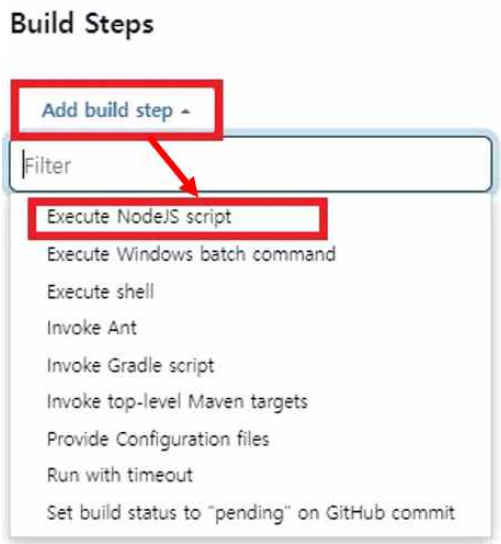


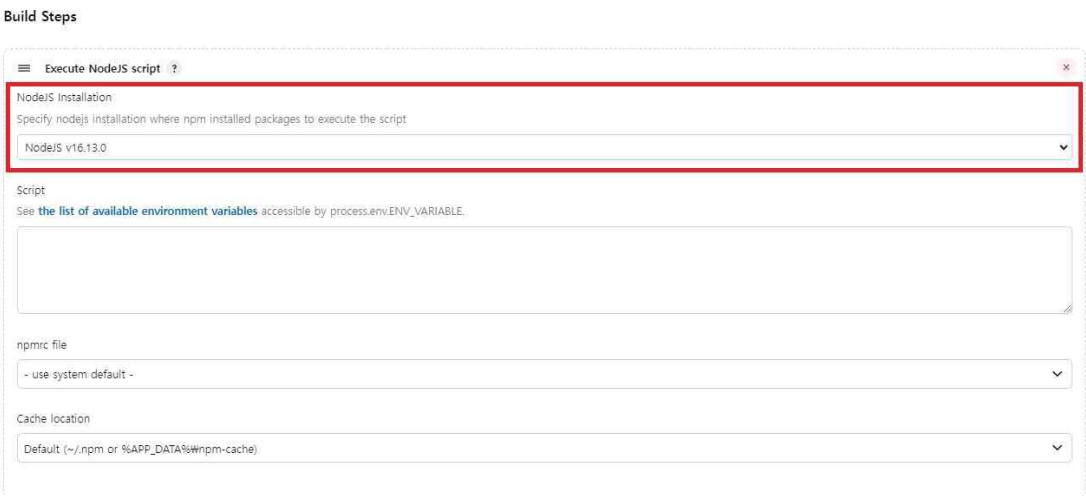
● 발급받은 Token 값을 복사하고 Add Jenkins의 password에 입력



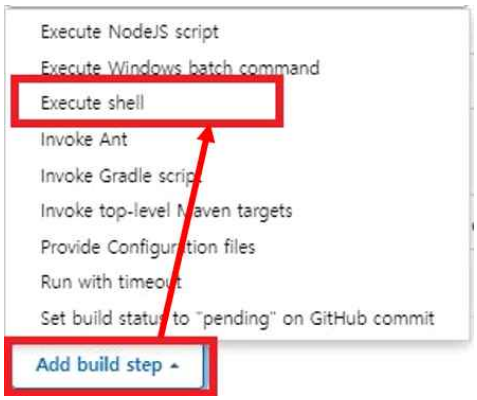
● Add Jenkins 설정을 마무리 한 다음, Credentials을 선택하고 Branches to build → Branch Specifier에서 main 브랜치를 입력

● Build Steps에서 Add build step → Execute NodeJS script 선택하고 앞에서 설정한 NodeJS를 선택





● Add build step → Execute shell 선택하고 다음과 같이 명령어 작성하고 저장



○ 빌드 전, Docker에 Jenkins 사용자 그룹 추가 진행하고 Jenkins 재시작

– 이 작업을 안할 시, 빌드 실행할 때 권한 문제로 실패

● sudo usermod –aG docker jenkins

● sudo service jenkins restart

```
student@example-server:~$ sudo usermod -aG docker jenkins
[sudo] password for student:
student@example-server:~$ sudo service jenkins restart
```

○ ‘지금 빌드’ 버튼을 눌러 빌드가 정상적으로 잘 되는지 확인