

Naver Cloud
Docker + Jenkins 자동배포
구현 매뉴얼

2023.06.27.

- 비트 마켓 -

변경내역

[illegible]

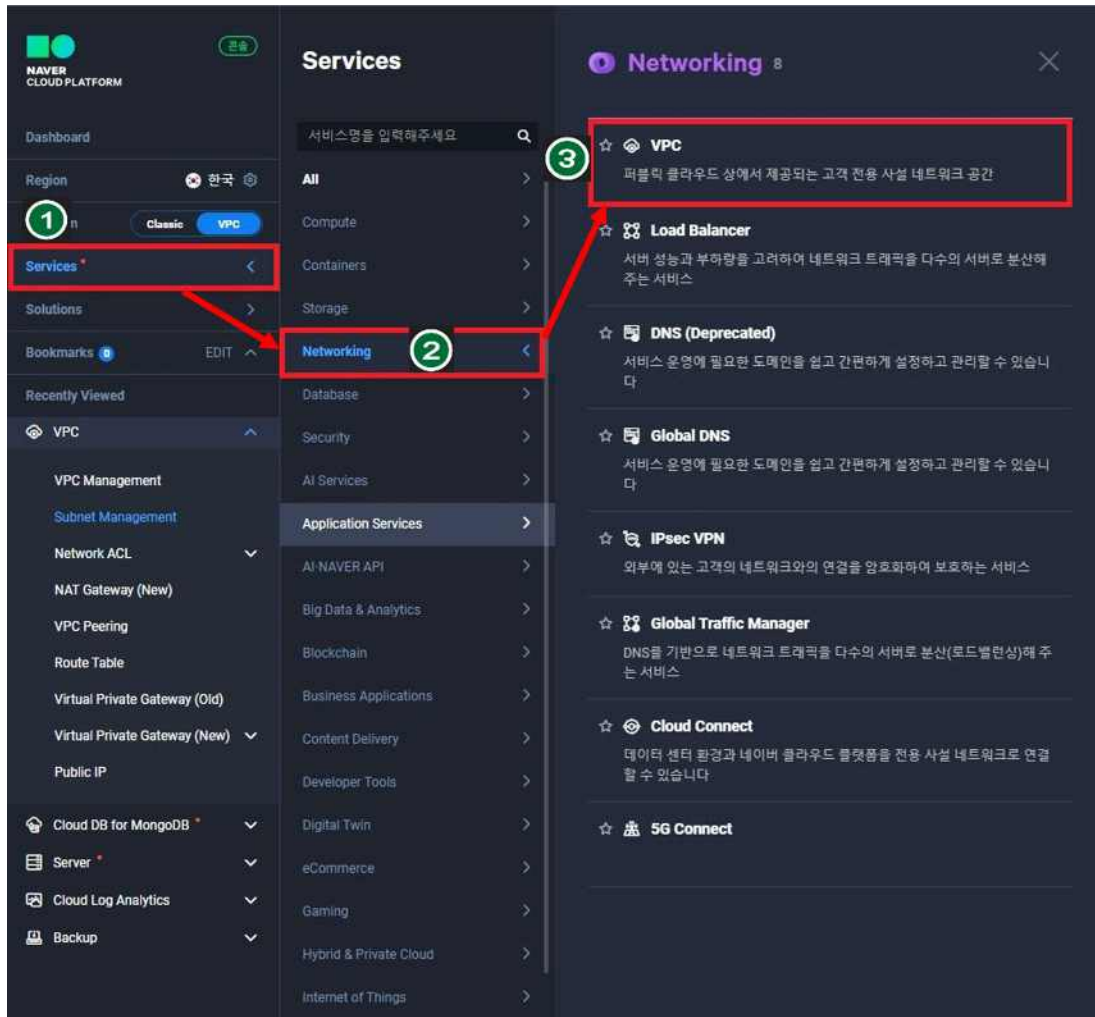
목 차

1. VPC 생성	4 ~ 5
2. Subnet 생성	6 ~ 7
3. Subnet 생성	8 ~ 11
4. Server Setting	12 ~ 14
5. Docker 설치	15 ~ 19
6. Jenkins 설치	20 ~ 26
7. Jenkins Build	27 ~ 34

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

VPC 생성 [X]

VPC를 생성합니다.

VPC는 논리적으로 격리된 네트워크 공간을 제공합니다.
VPC의 IP 주소 범위는, private 대역(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) 내에서 /16~/28 범위여야 합니다.

(필수 입력 사항입니다.)

VPC 이름: example-vpc

IP 주소 범위: 10.0.0.0/16

[X] 취소 [✓] **생성**

- 생성 후 상태가 운영중 인지 확인

VPC / VPC / VPC Management

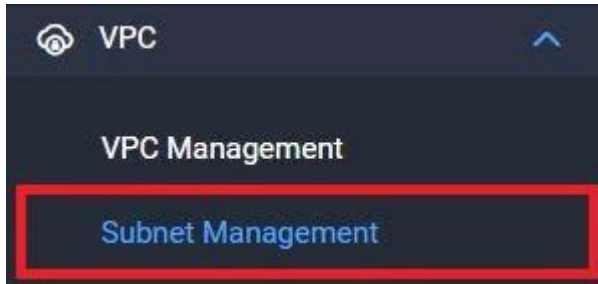
VPC (Virtual Private Cloud) 3

+ VPC 생성 새로고침

VPC 이름	VPC ID	상태	CIDR 블록
<input type="checkbox"/> example-vpc	39716	● 운영중	10.0.0.0/16

2 Subnet 생성

- 왼쪽 메뉴의 VPC에서 Subnet Management 클릭



- Subnet 생성 버튼 클릭

VPC / VPC / Subnet Management

Subnet 2



- 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

Subnet 생성

Subnet을 생성합니다.

VPC 내에 세분화된 격리 공간을 제공합니다.
IP 주소 범위는 VPC 주소 범위 이하로만 지정이 가능하며,
private 대역(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) 내에서 /16~/28 범위여야 합니다.
생성 이후 Network ACL 만 변경이 가능하므로 생성시 주의해주시기 바랍니다.

(필수 입력 사항입니다.)

Subnet 이름

example-pub

VPC

example-vpc (10.0.0.0/16)

생성한 리소스를 선택하려면 View/getVPCDetail 권한이 필요합니다.

IP 주소 범위

10.0.1.0/24

가용 Zone

KR-2

Network ACL

example-vpc-default-network-acl

생성한 리소스를 선택하려면 View/getNetworkACLDetail 권한이 필요합니다.

Internet Gateway 전용 여부

☒ Y (Public)
☐ N (Private)

용도

☒ 일반
☐ LoadBalancer
☐ BareMetal
☐ NatGateway

일반서버에서만 사용 가능한 서브넷입니다.

취소

생성

○ 생성 후 상태가 운영중 인지 확인

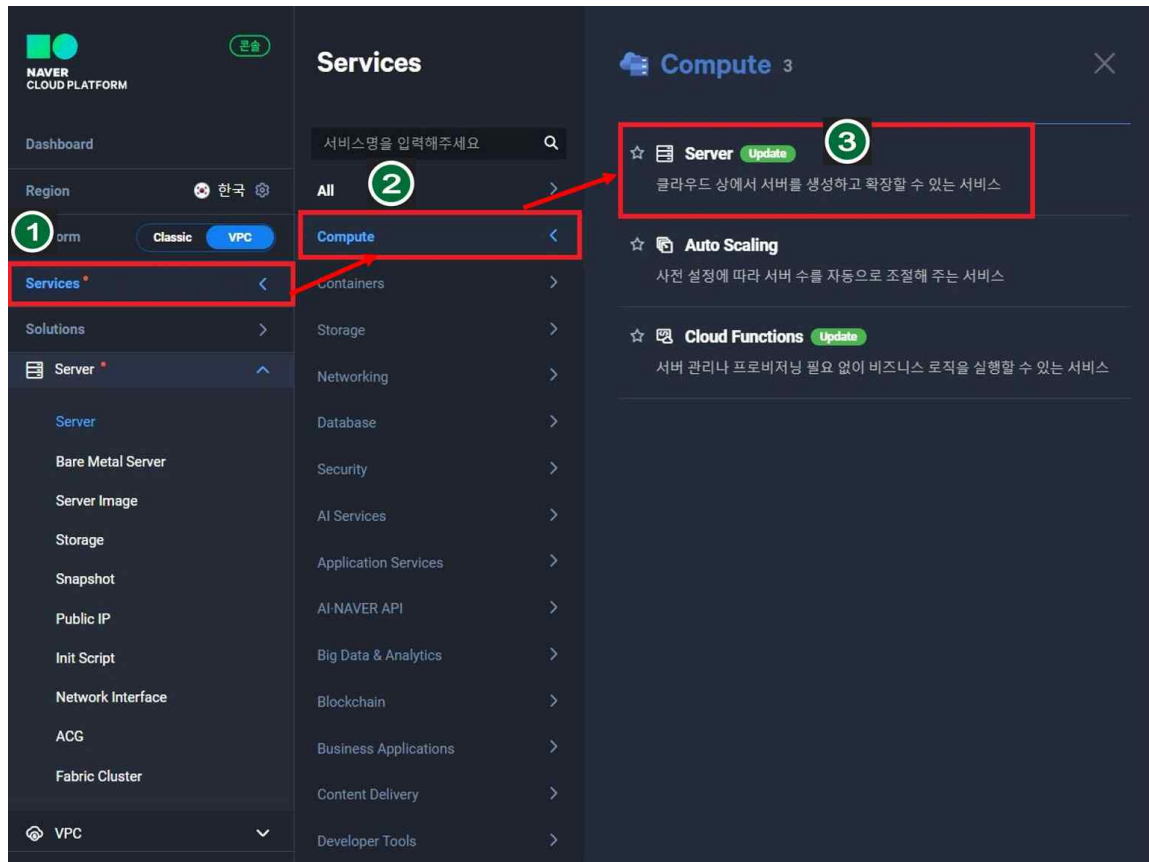
VPC / VPC / Subnet Management

Subnet 3

<div> <div>+ Subnet 생성</div> <div>새로고침</div> <div></div> </div>			
Subnet 삭제			
Subnet 이름	Subnet ID	상태	VPC 이름
<input type="checkbox"/> example-pub	87884	<input checked="" type="radio"/> 운영중	example-vpc

3 App Server 생성

○ 콘솔 창에서 Services → Compute → Server 클릭



○ Server 생성 버튼 클릭

VPC / Server / Server

Server 0

+ 서버 생성

상품 더 알아보기

X 다운로드

새로고침



○ 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 이미지 선택
CentOS, Ubuntu, Windows 및 DBMS 서버 이미지를 제공합니다. 이미지 및 부팅 디스크 크기를 선택하세요.
• Windows만 부팅 디스크로 100GB 선택이 가능합니다.

부팅 디스크 크기	<input checked="" type="radio"/> 50GB <input type="radio"/> 100GB
이미지타입	<input type="radio"/> Application <input type="radio"/> DBMS <input checked="" type="radio"/> OS
OS 이미지타입	<input type="radio"/> All <input type="radio"/> CentOS <input type="radio"/> Rocky <input checked="" type="radio"/> Ubuntu
서버 타입	<input checked="" type="radio"/> High CPU <input type="radio"/> Standard <input type="radio"/> High-Memory <input type="radio"/> GPU <input type="radio"/> CPU Intensive

서버 이미지 이름	설명	
ubuntu-16.04	Ubuntu Server 16.04 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)	다음 >
ubuntu-18.04	Ubuntu Server 18.04 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)	다음 >
ubuntu-20.04	Ubuntu Server 20.04 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)	다음 >

○ 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

서버 설정
서버 타입과 요금제를 선택하세요. (*필수 입력 사항입니다.)

VPC example-vpc VPC 생성

생성한 리소스를 선택하려면 View/getVPCDetail 권한이 필요합니다.

Subnet example-pub | KR-2 | 10.0.1.0/24 | Public Subnet 생성

공인 IP 연결을 위해서는 반드시 Public Subnet을 선택해야 합니다.
생성한 리소스를 선택하려면 View/getSubnetDetail 권한이 필요합니다.

스토리지 종류 ☒ SSD ☐ HDD

서버 세대 g2

서버 타입 High CPU

[High CPU] vCPU 2개, 메모리 4GB, [SSD]디스크 50GB [g2]

요금제 선택 ☐ 월요금제 ☒ 시간 요금제 시간 당 100 KRW (OS 제외)

서버 개수 1

서버 이름 example-server

☒ 입력하신 서버 이름으로 hostname을 설정합니다.

Network Interface

디바이스	Network Interface	Subnet	IP
eth1	new interface	- select -	미입력시 자동할당 + 추가
eth0	new interface	example-pub KR-2 10.0.1.0/24 Public	자동할당 ✖

공인 IP ☐ 미설정 ☒ 새로운 공인 IP 할당

- 신청된 공인 IP는 보유하신 동안 요금이 과금되므로, 사용하지 않을 때는 반납하시기를 권장드립니다. (월 이용료: 4,032원)
- 서버 생성시 공인 IP를 함께 생성하시려면 Subnet 타입은 Public Subnet, 서버 개수는 1개여야 합니다.

플리 배지 그룹 ☒ 미사용 ☐ 사용

반납 보호 ☐ 설정 ☒ 해제

반납 보호를 설정하면 실수로 반납하는 사고를 방지할 수 있습니다.

메모 여 세기를 입력하세요

Script 선택 선택없음

○ 인증키 설정에서 선택 후 다음 버튼 클릭

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

인증키 설정

보유하고 있는 인증키를 선택하거나 새로운 인증키를 생성하세요. 인증키는 관리자 비밀번호를 얻는데 사용됩니다. (*필수 입력 사항입니다.)

☒ 보유하고 있는 인증키 이용

인증키 선택 customer

☐ 새로운 인증키 생성

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

인증키 설정

보유하고 있는 인증키를 선택하거나 새로운 인증키를 생성하세요. 인증키는 관리자 비밀번호를 얻는데 사용됩니다. (*필수 입력 사항입니다.)

☐ 보유하고 있는 인증키 이용

☒ 새로운 인증키 생성

인증키 이름 example 인증키 생성 및 저장

인증키 이름을 입력 후 [인증키 생성 및 저장]를 클릭하여 인증키를 사용자 컴퓨터에 저장하세요.
인증키는 해당 서버의 관리자 비밀번호 확인에 이용되니 안전한 곳에 저장하시기 바랍니다

○ 네트워크 접근 설정 단계에서 eth0 선택 후 다음 버튼 클릭

- eth0은 기본 값을 선택하였음

네트워크 접근 설정

보유하고 있는 ACG를 선택하거나 새로운 ACG를 생성해주세요. (*필수 입력 사항입니다.)
ACG(Access Control Group)은 별도의 방화벽 구축없이, 서버 그룹에 대한 네트워크 접근 제어 및 관리를 돕는 상품입니다.

디바이스	ACG
eth0	example-vpc-default-acg x

최대 3개까지 선택가능
생성한 리소스를 선택하려면 View/getACGDetail 권한이 필요합니다.

설정 시 주의사항

- '신규 Network Interface'로 지정해서 생성된 ACG만 설정 가능합니다.
- 기존에 생성된 'Network Interface'를 사용하는 경우 해당 Network Interface의 ACG를 사용하게 됩니다.

○ 최종 단계에서 설정한 값들이 맞는지 확인 후 생성 버튼 클릭

- 생성 시간은 최장 약 30분 정도 소요

최종 확인
[서버 생성] 버튼을 클릭하면 서버가 생성됩니다.

서버 이미지

서버 이미지 이름	서버 이미지 설명
ubuntu-20.04	Ubuntu Server 20.04 (64-bit)

서버

스토리지 종류	SSD	서버 이름	example-server
서버 타입	[High CPU] vCPU 2개, 메모리 4GB, [SSD]디스크 50GB [g2]	요금제	시간 요금제
Region		기본 스토리지 암호화 적용	N
메모		Zone	KR-2
플리 태지 그룹	미사용중	반납 보호	해제

네트워크

VPC	Subnet
example-vpc	example-pub KR-2

인증키

인증키 이름
example

Access Control Group

Device	Access Control Group
eth0	example-vpc-default-acg

Script 선택

선택없음 상세내용

4 Server 설정

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

서버 관리 및 설정 변경

강제 반납

1 관리

관리자 비밀번호 확인 (000583)

서버 접속에 필요한 비밀번호를 확인합니다.

유사 서버 생성

스토리지 생성

서버 설정 변경

반납 보호 설정 변경

공인 IP 설정 변경

Secondary IP 설정 변경

Network 모니터링 설정 변경

상세 모니터링 설정 변경

서버 스택 변경

물리 배치 그룹 변경

Network Interface 할당

관리자 비밀번호 확인

인증키 내용을 확인합니다.

관리자 비밀번호를 확인하기 위해서 해당 서버의 인증키가 필요합니다.
서버 생성 시에 설정한 인증키 파일을 첨부하시고, [비밀번호 확인]을 클릭하면
관리자 비밀번호가 제공됩니다.

2

(• 필수 입력 사항입니다.)

서버 이름 example-server

인증키 이름* example

+ 마우스로 파일을 끌고 오거나 여기를 클릭하세요

(ex) C:\Users\<사용자명>\Downloads\theauth.pem

× 취소 ✓ 비밀번호 확인

- 관리자 이름과 비밀번호를 확인

관리자 비밀번호

최초 생성시에 제공되는 관리자 비밀번호입니다.

서버에 접속한 후 고객님의 기억할 수 있는 비밀번호로
변경하실 것을 권장합니다.

※ 추측하기 쉬운 패스워드는 해킹 가능성이 있어 안전한 패스워드 사용
(8자 이상의 알파벳 대소문자/숫자/특수문자를 조합)을 권고 드립니다.

서버 이름 example-server

관리자 이름 root

비밀번호

확인

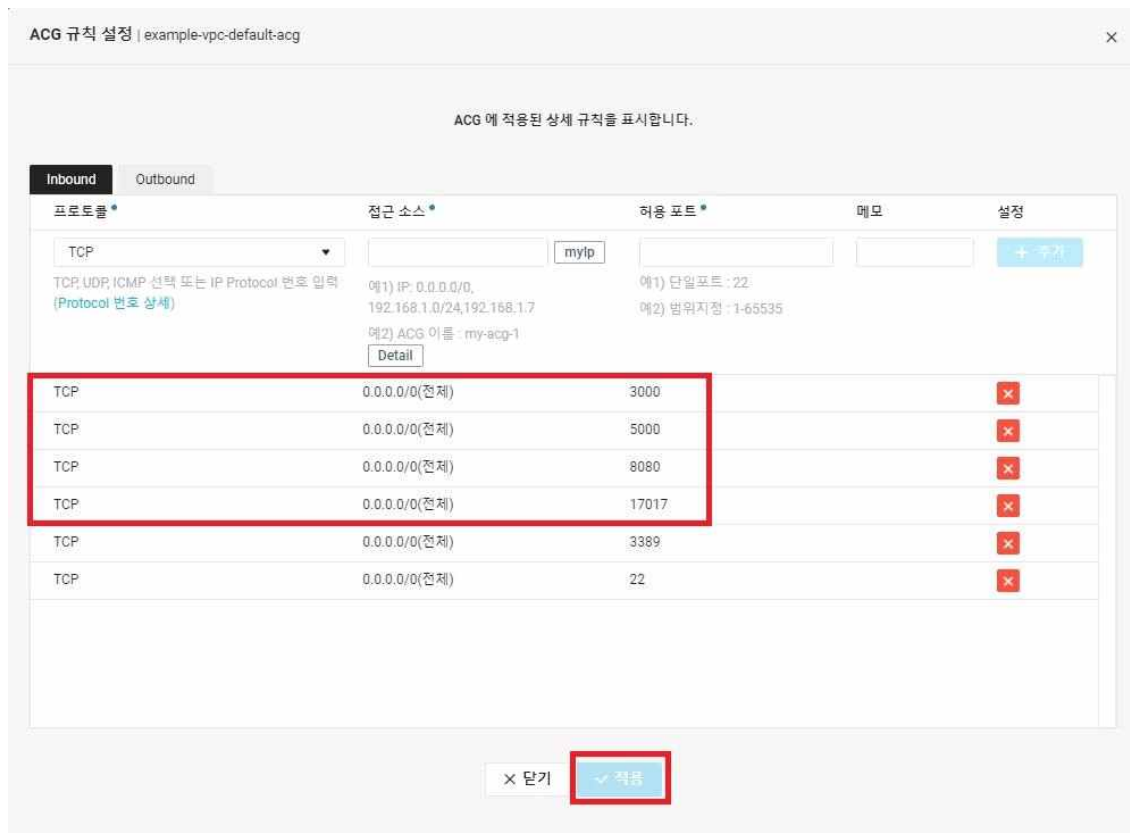
○ 콘솔에서 Server → ACG → ACG 설정 클릭

– 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

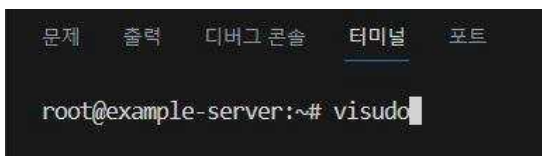
- 여기서는 계정을 student로 생성하였음

● 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 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 slurp4netns
Suggested packages:
  aufs-tools cgroups-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 slurp4netns
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.

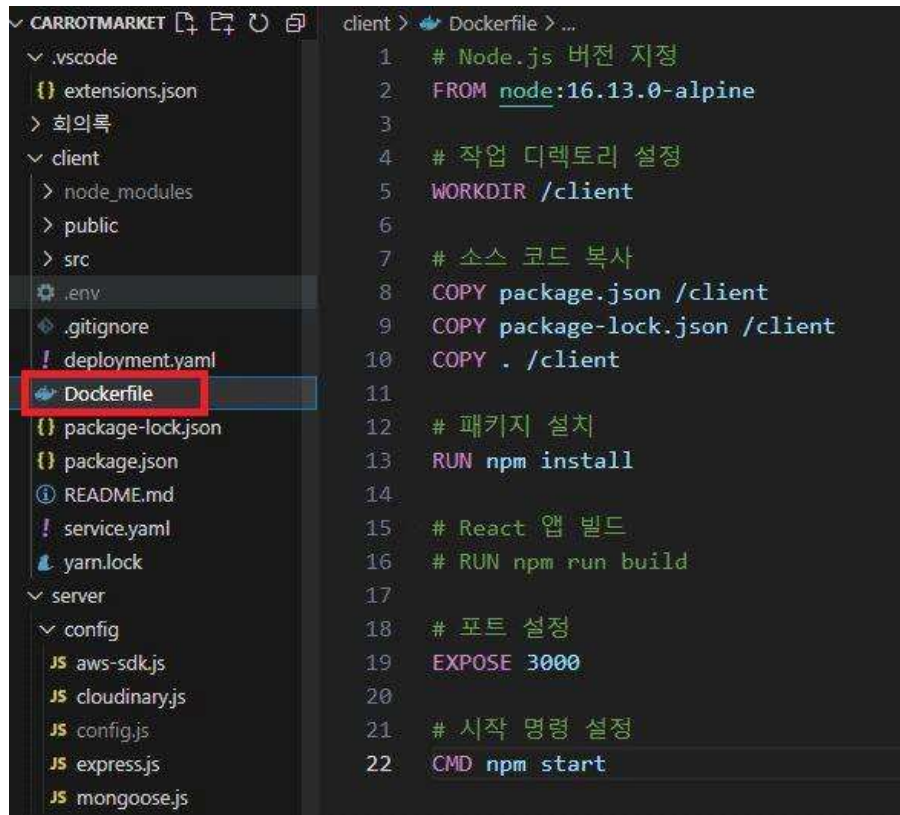
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

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/
```

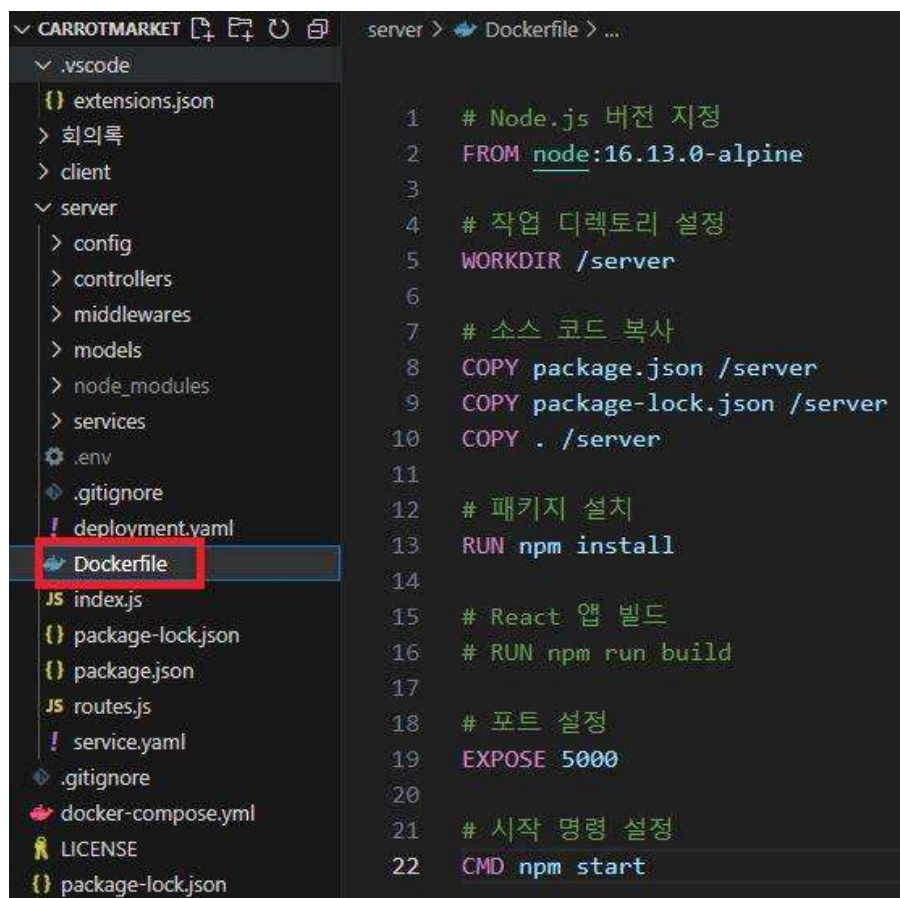
○ 프로젝트 내 client · server 폴더 내부에 다음과 같이 Dockerfile 작성

● client 내부 Dockerfile



```
client > Dockerfile > ...
1  # Node.js 버전 지정
2  FROM node:16.13.0-alpine
3
4  # 작업 디렉토리 설정
5  WORKDIR /client
6
7  # 소스 코드 복사
8  COPY package.json /client
9  COPY package-lock.json /client
10 COPY . /client
11
12 # 패키지 설치
13 RUN npm install
14
15 # React 앱 빌드
16 # RUN npm run build
17
18 # 포트 설정
19 EXPOSE 3000
20
21 # 시작 명령 설정
22 CMD npm start
```

● server 내부 Dockerfile



```
server > Dockerfile > ...
1  # Node.js 버전 지정
2  FROM node:16.13.0-alpine
3
4  # 작업 디렉토리 설정
5  WORKDIR /server
6
7  # 소스 코드 복사
8  COPY package.json /server
9  COPY package-lock.json /server
10 COPY . /server
11
12 # 패키지 설치
13 RUN npm install
14
15 # React 앱 빌드
16 # RUN npm run build
17
18 # 포트 설정
19 EXPOSE 5000
20
21 # 시작 명령 설정
22 CMD npm start
```


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:
```

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:

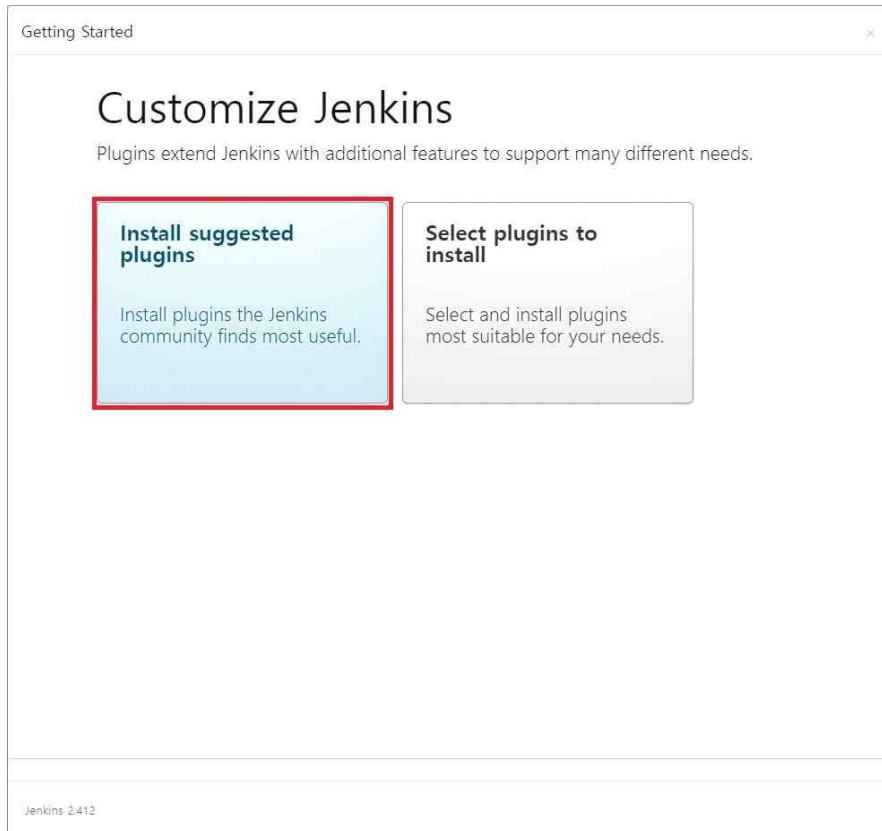
```
/var/lib/jenkins/secrets/initialAdminPassword
```

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

Administrator password

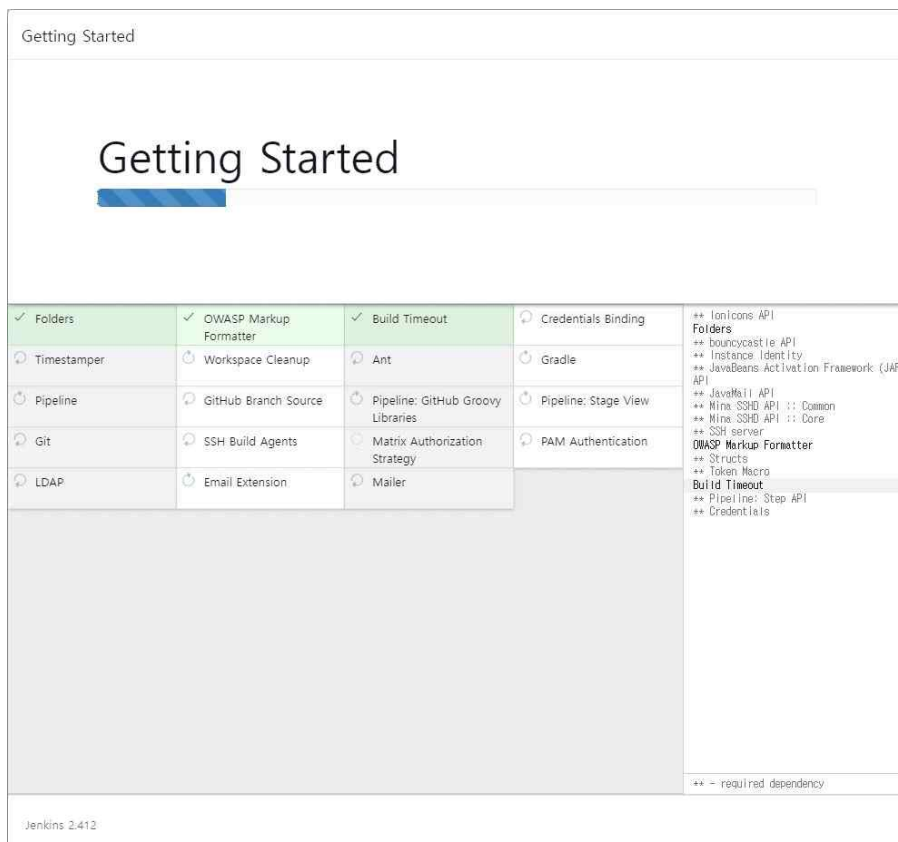
Continue

○ Customize Jenkins에서 Install suggested plugins 선택



○ 설치가 진행되고 끝나면 Create First Admin User에서 계정 설정 진행

– 계정은 예시로 설정하였음



Getting Started

Create First Admin User

계정명
tjchswkd

암호

암호 확인

이름
EOM TAEHUN

이메일 주소
tjchswkd@naver.com

Jenkins 2.412

[Skip and continue as admin](#) [Save and Continue](#)

- Instance Configuration에서 Jenkins URL 설정
- 여기서는 Jenkins URL을 default 값으로 설정하였음

Getting Started

Instance Configuration

Jenkins URL:

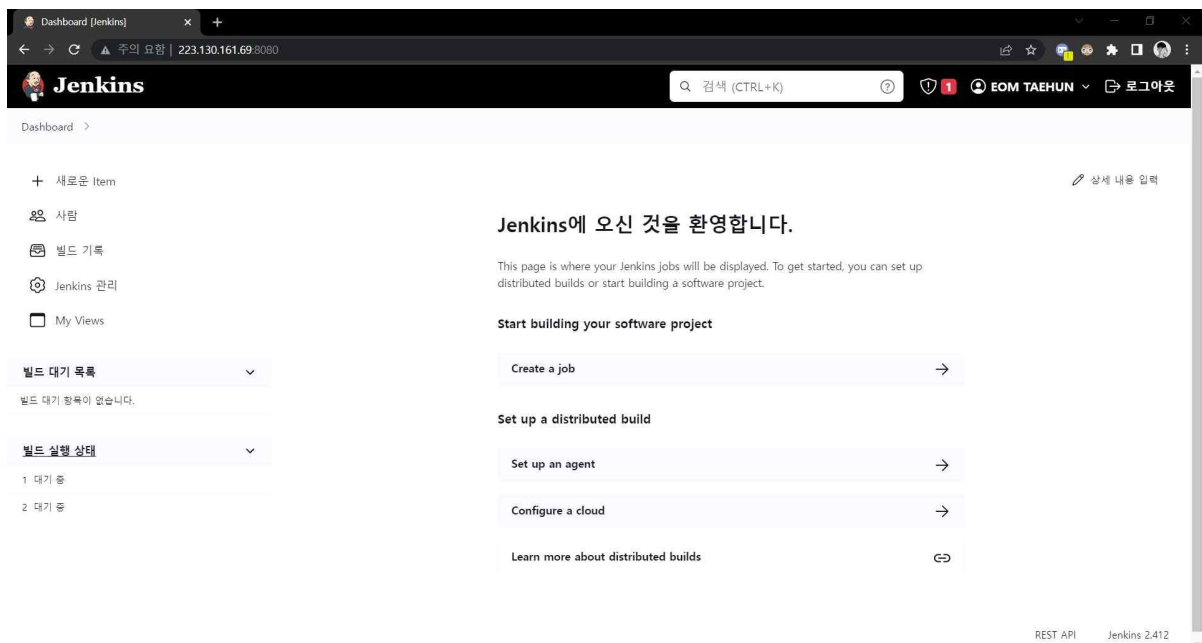
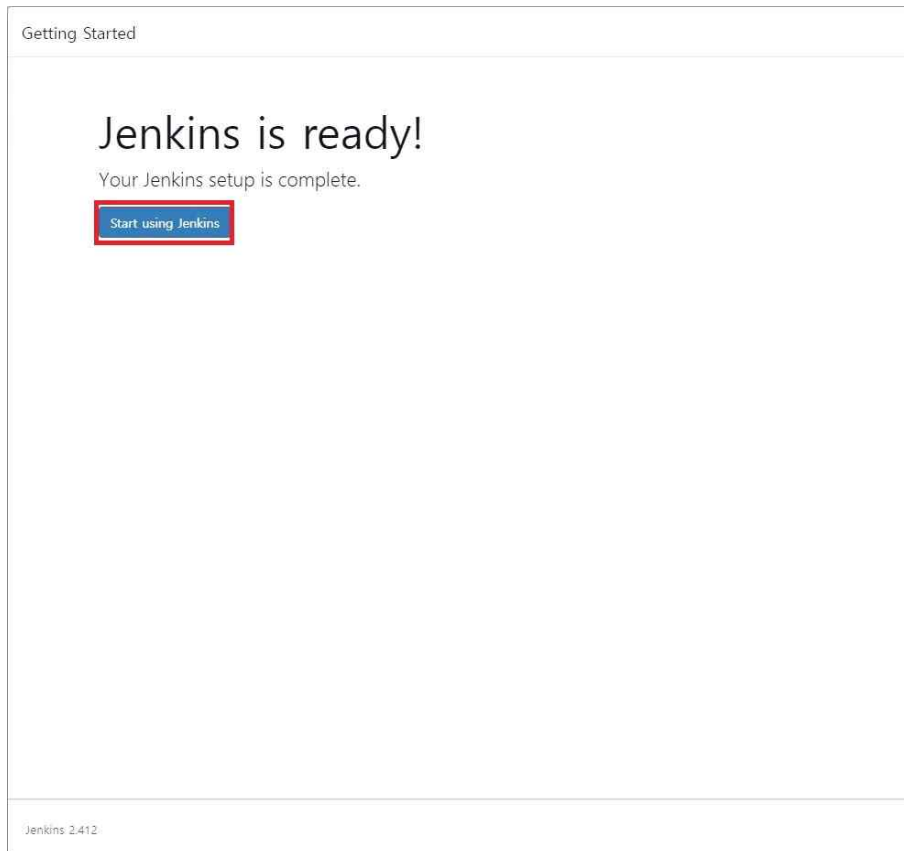
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

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 2.412

[Not now](#) [Save and Finish](#)

- 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


The screenshot shows the Jenkins management interface. On the left sidebar, 'Jenkins 관리' is selected. In the main area, 'System Configuration' is shown with tabs for System, Tools, and Plugins. The 'Plugins' tab is selected, and a search for 'node' is performed. The 'Available plugins' list shows 'NodeJS 1.6.0' selected. Below the list, the 'Download now and install after restart' button is highlighted. A red box highlights the 'Available plugins' section in the sidebar and the 'NodeJS 1.6.0' entry in the list. A red arrow points from the 'Jenkins 관리' menu item to the 'Plugins' tab. Another red arrow points from the 'Available plugins' section in the sidebar to the 'NodeJS 1.6.0' entry. A third red arrow points from the 'Download now and install after restart' button to the 'NodeJS 1.6.0' entry.

○ Download progress에서 ‘설치가 끝나고 실행중인 작업이 없으면 Jenkins 재시작’ 체크

Download progress

준비

- Checking internet connectivity
- Checking update center connectivity
- Success

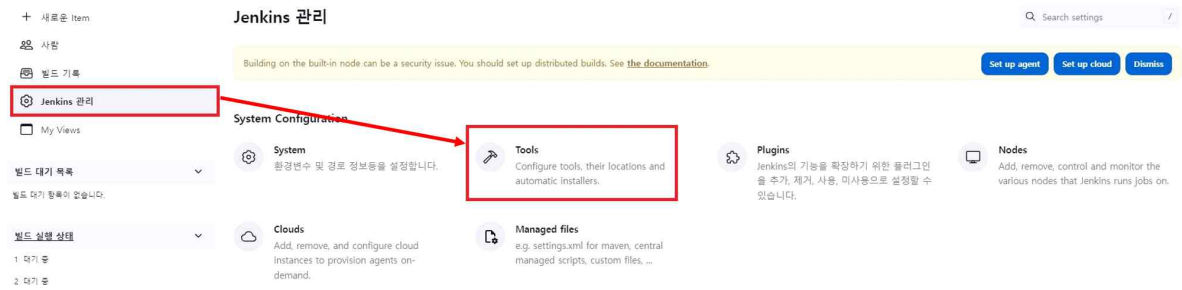
NodeJS  Downloaded Successfully. Will be activated during the next boot

→ [메인 페이지로 돌아가기](#)

(설치된 플러그인을 바로 사용하실 수 있습니다.)

→ ☐ 설치가 끝나고 실행중인 작업이 없으면 Jenkins 재시작.

○ 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

Gradle installations

Add Gradle

Ant installations

Add Ant

Maven installations

Add Maven

NodeJS installations

Add NodeJS

A screenshot of the 'NodeJS' configuration form in Jenkins. The form is titled 'NodeJS' and has several fields. The 'Name' field is set to 'NodeJS v16.13.0'. The 'Install automatically' checkbox is checked. Under the 'Install from nodejs.org' section, the 'Version' field is set to 'NodeJS 16.13.0'. There are also fields for 'Global npm packages to install', 'Global npm packages refresh hours', and 'Force 32bit architecture'. The 'Add Installer' button is at the bottom of the form.

Add NodeJS

Save

Apply

○ Jenkins 메인 화면 → Create a job 클릭

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 ↗


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


– item name은 예시로 example로 설정하였음


Enter an item name


example
Required field


**Freestyle project**
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

OK

○ General에서 각각의 항목들에 대해 다음과 같이 설정

● GitHub project: 프로젝트의 GitHub 주소

GitHub project

Project url ?

https://github.com/ksqrt/carrotMarket.git

고급 ▾

☐ Throttle builds ?

☐ 오래된 빌드 삭제 ?

☐ 이 빌드는 매개변수가 있습니다 ?

☐ 필요한 경우 concurrent 빌드 실행 ?

고급 ▾

● 소스코드 관리 → Git → Add Jenkins

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/ksqrt/carrotMarket.git

Credentials ?

-- none - ▾

Add +

Jenkins

고급 ▾

Add Repository

● Add Jenkins 설정

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted) ▾

Kind

Username with password ▾

Scope ?

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

Username ?

TH1117

☐ Treat username as secret ?

Password ?

ID ?

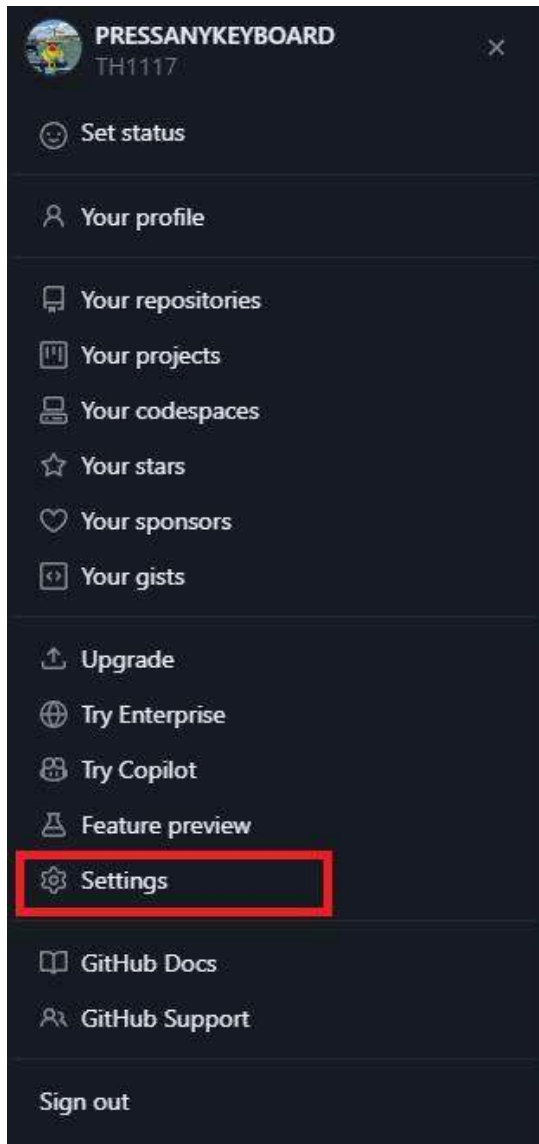
PRESSANYKEYBOARD

Description ?

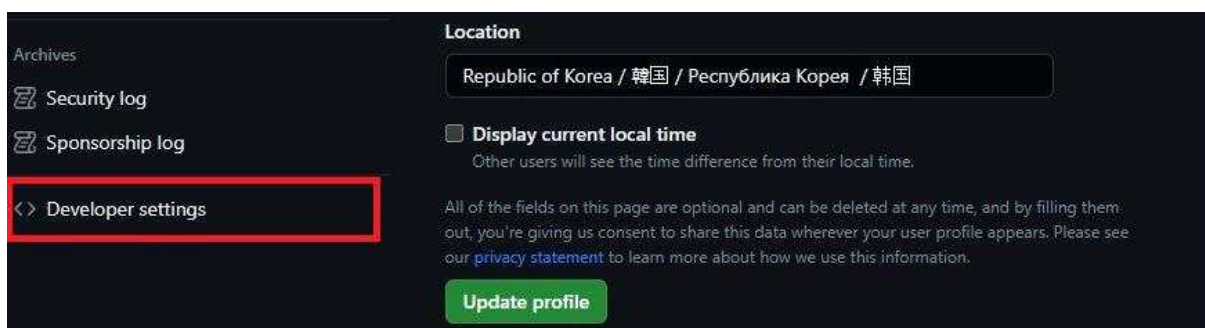
Add Cancel

※ Add Jenkins의 Password 입력 방법

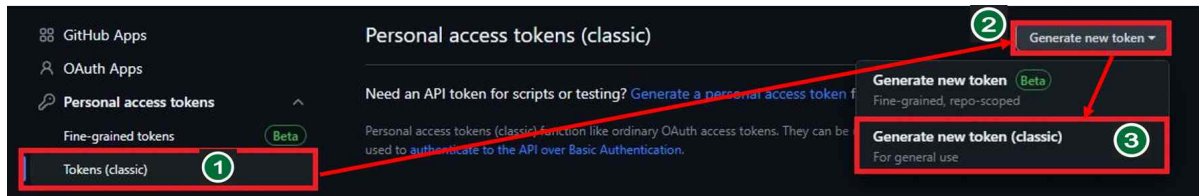
- 자기자신의 GitHub 프로필 → Settings 클릭



- 좌측 하단의 Developer settings 클릭



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



- Note, Expiration을 설정하고 Select scopes에서 repo만 체크하고 Generate token 클릭

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

example

What's this token for?

Expiration *

No expiration ⌵ The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

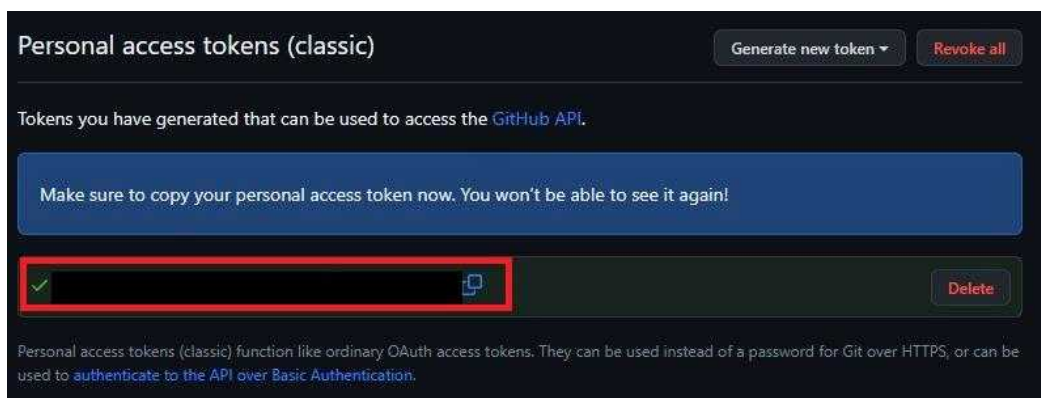
Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

Generate token Cancel

- 발급받은 Token 값을 복사하고 Add Jenkins의 password에 입력



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

The screenshot shows the Jenkins configuration page for a new job. The 'Repository URL' field is set to 'https://github.com/ksqrt/carrotMarket.git'. The 'Credentials' dropdown menu is open, showing 'TH1117/*****' selected. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field set to '*/main'. Red boxes highlight the 'Credentials' and 'Branch Specifier' fields.

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

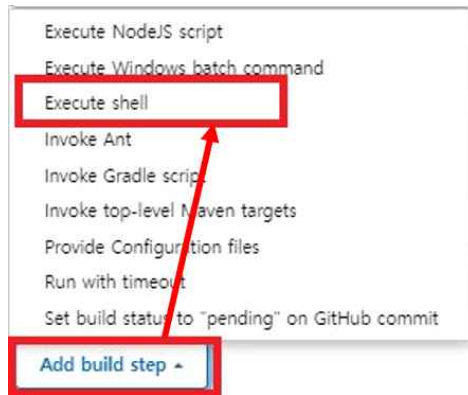
Build Steps

The screenshot shows the 'Add build step' dropdown menu. The 'Add build step' button is highlighted with a red box. The dropdown menu is open, showing a list of build steps. 'Execute NodeJS script' is highlighted with a red box. Other options include 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Provide Configuration files', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'.

Build Steps

The screenshot shows the configuration page for the 'Execute NodeJS script' build step. The 'NodeJS Installation' dropdown menu is open, showing 'NodeJS v16.13.0' selected. The 'Script' field is empty. The 'npmrc file' dropdown menu is set to 'use system default'. The 'Cache location' dropdown menu is set to 'Default (-/./npm or %APP_DATA%\npm-cache)'. Red boxes highlight the 'NodeJS Installation' and 'Script' fields.

- 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

