

포팅메뉴얼

1. 사용 버전
2. 빌드
3. 배포
4. DB

사용 버전

프로젝트 사용 도구

- 이슈관리 : Jira
- 형상관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- UI/UX : Pigma

프로젝트 개발 환경

Frontend

- Visual Studio Code : 1.70.0
- react : 18.2.0
- react-dom : 18.2.0
- Node.js : 16.17.1

Backend

- IntelliJ : 11.0.15+10-b2043.56 amd64
- Java : 11
- SpringBoot : 2.7.5

DB

- MySQL : 8.0.29

Server

- Ubuntu : 20.04

빌드

프로젝트 빌드방법

Frontend

```
npm i
npm start
```

Backend

```
Gradle -> build
```

프로젝트 환경변수

Frontend

```
REACT_APP_API_URL={base url}
REACT_APP_AWS_ACCESSKEY={aws access key}
REACT_APP_AWS_SECRETACCESSKEY={aws secret key}
```

Backend

```
server:
  port: 포트번호
  error:
    whitelabel:
      enabled: false

  servlet:
    context-path: /api

spring:
  # TODO : PUSH 하기 전에 다시 한번 확인할 것.
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ec2 mysql 주소
    username: 유저이름
```

```

password: 비밀번호
mvc:
  pathmatch:
    matching-strategy: ant_path_matcher

jpa:
  database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
  open-in-view: false
  hibernate:
    format_sql: true
    ddl-auto: update
    show-sql: true

logging:
  level:
  org:
    hibernate:
      SQL: DEBUG
    type:
      descriptor:
        sql:
          BasicBinder: TRACE
  com:
    amazonaws:
      util:
        EC2MetadataUtils: error

cloud:
  aws:
    credentials:
      accessKey: aws accesskey
      secretKey: aws secretkey
    s3:
      bucket: dyk
      path: s3 path
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false

```

GIT Ignore

Frontend

```
.env.local
```

Backend

```
application.yml
```

배포

목차

1. Jenkins/Docker/Nginx 설치
2. SSL 인증서 적용
3. BackEnd - API 서버(Spring Boot)
4. FrontEnd - 가이드 페이지
5. FrontEnd - MyINI App
6. 소셜 로그인을 위한 nginx 설정

1. Jenkins/Docker/Nginx 설치

1.1. Jenkins 서버 접속

- window에서는 푸티
- mac에서는 ssh 지원 → 리눅스 서버로 접속 가능
- 권한 변경: `chmod 400 XXXXXXXX.pem`
- ssh명령어를 입력해서 서버의 원격으로 접속함: `ssh -i XXXXXXXX.pem ubuntu@i7b306.p.ssafy.io`
→ 서버의 원격으로 접속 완료

1.2. JAVA 설치 확인

`java -version`

→ java version 확인

1.3. 서버의 Timezone 변경

```
sudo rm /etc/localtimesudo ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtimedate
```

1.4. Ubuntu Docker 설치

```
sudo apt updatesudo apt install apt-transport-https ca-certificates curl software-properties-commoncurl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
```

```
bionic stable"sudo apt updateapt-cache policy docker-cesudo apt install docker-cesudo  
systemctl status docker (control+C)sudo usermod -aG docker ubuntu  
sudo chmod 666 /var/run/docker.sock
```

1.5. Jenkins-Docker 연결

1. DooD 방식

```
# Dockerfile 생성  
sudo vim Dockerfile
```

```
이미지를 만들기 위한 도커 파일  
FROM jenkins/jenkins:jdk11  
  
USER root  
  
COPY docker_install.sh /docker_install.sh  
RUN chmod +x /docker_install.sh  
RUN /docker_install.sh  
  
RUN usermod -aG docker jenkins  
USER jenkins
```

1. `docker_install.sh` 작성

```
sudo vim docker_install.sh
```

```
#!/bin/sh  
apt-get update && \apt-get -y install apt-transport-https \ ca-certificates  
 \ curl \ gnupg2 \ zip \ unzip \ software-properties-common && \curl -fsSL http  
s://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-ke  
y 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
```

1. docker.sock 파일의 권한 변경**

```
sudo chmod 666 /var/run/docker.sock
```

1. 이미지 생성

```
docker build -t jenkins .
```

1. 볼륨 마운트할 폴더 생성

```
mkdir jenkins sudo chown -R 1000 ./jenkins
```

1. Jenkins 실행

```
sudo docker run -d --name jenkins -v /home/ubuntu/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -p 8080:8080 -e TZ=Asia/Seoul jenkins
```

1.6. Nginx

1. Nginx 설치

```
sudo apt-get updatesudo apt install nginx -y sudo service nginx status
```

1. `/etc/nginx/sites-available` 아래 `test.conf` 설정파일 만들기

```
server { listen 80; server_name example.com www.example.com; return 301 https://example.com$request_uri;}server { server_name example.com www.example.com; location / { proxy_pass http://localhost:8080; proxy_set_header Host $http_host; proxy_set_header X-Real-IP $remote_addr; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; }}
```

1. Nginx 재구동/재시작

```
sudo service nginx reload # 재구동 -> 적용sudo service nginx restart # 위의 명령어가 적용안될 시 재시작하기
```

2. SSL 인증서 적용

2.1. 개요

- 백엔드 애플리케이션 (스프링부트)와 Openvidu 서버의 https 인증을 설정하기 위한 Let's Encrypt SSL 인증서 발급 및 적용 절차
- `Standalone` 방식의 SSL 인증서 발급
 - 사이트를 멈추고 사이트의 네크워킹을 이용해 사이트 유효성 확인
 - 80포트로 가상 standalone 웹서버를 띄워 인증서를 발급

- **장점** : 여러 도메인 발급 가능, ssl 인증서 발급 방식 중 상대적으로 안정적인 방식
- **단점** : 인증서 발급 시 Nginx를 중단하고, 발급 완료 후 다시 시작해야 함 ⇒ 인증서 발급 중 서비스 중단
- 주의할 점
 - 발급 전 반드시 80포트의 프로세스를 중지할 것!!
 - 중지해도 80포트가 비어있지 않거나 인증 절차에 문제가 생긴다면? ⇒ **해당 포트의 프로세스 강제로 죽이기**

2.1. 절차

1. Certbot 및 letsencrypt 설치

```
sudo apt update# Ubuntu 20.04 이후에는 letsencrypt 설치 시 certbot이 포함되어 있기 때문에 따로 설치할 필요가 없음!sudo apt-get install letsencrypt -y # letsencrypt 설치
```

1. nginx 서버 중단

```
sudo systemctl stop nginx
```

1. certbot 명령을 이용한 SSL 인증 (standalone)

```
certbot certonly --standalone -d [도메인 이름]# certbot certonly --standalone -d i7b306.p.ssafy.io### 설치 ...### Congratulations! 메시지와 함께 인증서와 체인 위치 출력
```

1. `/etc/nginx/sites-available` 로 이동하여 `test.conf` 수정

- test.conf 이름은 확장자만 동일하면 자유롭게 지정해도 상관없다.

```
# ...server {    listen 443 ssl http2; # https 포트    listen [::]:443;    server_name    i7b306.p.ssafy.io; # 도메인 이름    # SSL 인증서 적용    ssl_certificate /etc/letsencrypt/live/i7b306.p.ssafy.io/fullchain.pem;    ssl_certificate_key /etc/letsencrypt/live/i7b306.p.ssafy.io/privkey.pem;    # ...}
```

1. 심볼릭 링크 설정

```
sudo ln -s /etc/nginx/sites-available/test.conf /etc/nginx/sites-enabled/test.conf
```

1. Nginx 재시작

```
sudo nginx -t # nginx test. success가 뜨면 nginx 실행 가능  
sudo systemctl restart nginx
```

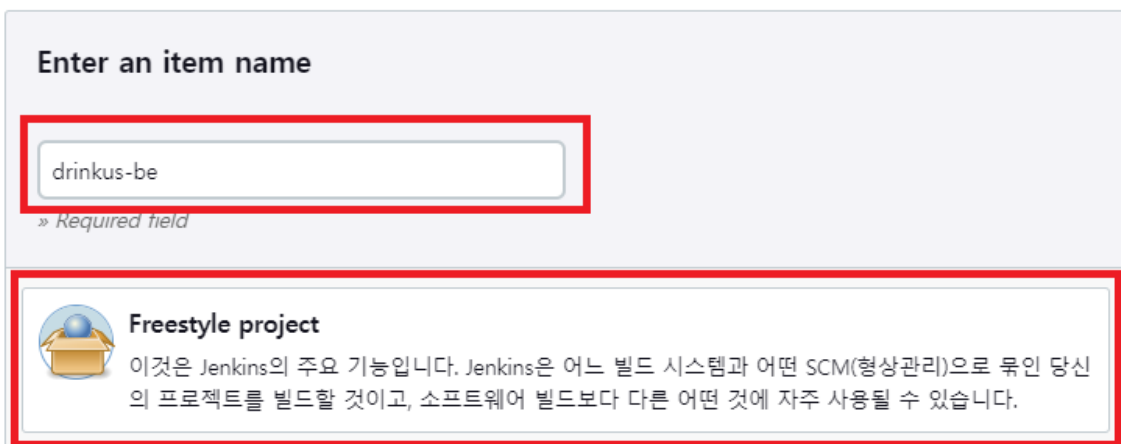
2.3. 결과

- http로 80포트 접근 시 https 443포트로 리다이렉트
- `https://도메인` 주소로 배포한 웹페이지에 접근 가능

3. BackEnd - API 서버(Spring Boot)

3.1. Jenkins 프로젝트 생성


1. **새로운 Item** 에서 새 프로젝트를 생성
프로젝트명을 입력하고 Freestyle project 선택



Enter an item name

drinkus-be

» 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.

2. **프로젝트-구성** 에서 프로젝트 설정 정보 구성

Git ?

Repositories ?

Repository URL ?

깃허브 프로젝트 URL

Credentials ?

이전에 생성했던 Credentials

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/BE

Add Branch

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://i7b306.p.ssafy.io:8080/project/drinkus-backend> ?
 - Enabled GitLab triggers
 - ☒ Push Events
 - ☐ Push Events in case of branch delete
 - ☐ Opened Merge Request Events
 - ☐ Build only if new commits were pushed to Merge Request ?
 - ☐ Accepted Merge Request Events
 - ☐ Closed Merge Request Events

빌드 환경

☐ Delete workspace before build starts

☒ Use secret text(s) or file(s) ?

Bindings

Username and password (separated) ?

Username Variable ?

USERNAME

Password Variable ?

PASSWORD

Credentials ?

☒ Specific credentials ☐ Parameter expression

credentials (dockerhub account)

+ Add

Add ▾

☒ SSH Agent

Credentials

☒ Specific credentials ☐ Parameter expression

ubuntu (server account)

+ Add

추가

☐ Ignore missing credentials

1. secret token 발급 Jenkins Project - 구성 - 빌드 유발 - 고급 Generate 를 누르면 Secret Token 이 발급됩니다. 이후 GitLab과 연동 시 사용

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)

☐ Build on successful pipeline events

Pending build name for pipeline [?](#)

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job [?](#)

☐ Filter branches by name [?](#)

☐ Filter branches by regex [?](#)

☐ Filter merge request by label

Secret token [?](#)

[Generate](#)

2. 빌드 스크립트 Execute Shell

```
pwdlscd backchmod 700 ./gradlew./gradlew clean buildcd myini-apilsdocker build -t gkse
kqls9808/myini-api .echo $PASSWORD | docker login -u $USERNAME --password-stdindocker
push gkseqls9808/myini-apidocker rmi gkseqls9808/myini-apipwdcd ..cd scriptspwdlsss
h -o StrictHostKeyChecking=no ubuntu@k7b203.p.ssafy.io -t < /var/jenkins_home/workspac
e/back/back/scripts/deploy.sh
```

Google Vision API 사용자 인증 키를 도커 컨테이너로 복사하기 위한 코드가 추가되었다

빌드 후 조치

☰ Delete workspace when build is done ✕

[고급...](#)

image

3.2. GitLab 연동

1. network 설정

2. Settings - Webhooks

확인 및 push 테스트 가능
Jenkins에서 빌드 내역 확인 가능

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

서버url:port/project/프로젝트명

URL must be percent-encoded if it contains one or more special characters.

Secret token

발급받은 secret token

Used to validate received payloads. Sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☒ Push events

Branch name or wildcard pattern to trigger on (leave blank for all)

Push to the repository.

SSL verification

☒ Enable SSL verification

Add webhook

Project Hooks (3)

http:// [redacted] [Test] [Edit] [Delete]

Push Events SSL Verification: enabled Push events

✓ #166 2022. 8. 18. 오전 11:05

Started by GitLab push by [redacted]

3.3. Dockerfile 및 script

1. api 프로젝트 내부에 Dockerfile 작성

```
FROM openjdk:17-ea-11-jdk-slimVOLUME /tmpCOPY build/libs/myini-api-0.0.1-SNAPSHOT.jar myini-api.jarENTRYPOINT ["java", "-jar", "myini-api.jar"]
```

1. project root - scripts 에 deploy.sh 작성

```
#!/usr/bin/env bashecho $PASSWORD | docker login -u $USERNAME --password-stdinRESPONSE_CODE=$(curl -s -o /dev/null -w "%{http_code}" https://k7b203.p.ssafy.io/api/port)echo "> 응답 코드 $RESPONSE_CODE"if [ $RESPONSE_CODE -ge 400 ]then CURRENT_PORT=8082else CURRENT_PORT=$(curl -s https://k7b203.p.ssafy.io/api/port)fi echo "> 현재 구동중인 포트 $CU
```

```

CURRENT_PORT"if [ ${CURRENT_PORT} -eq 8081 ]then IDLE_PORT=8082elif [ ${CURRENT_PORT}
-eq 8082 ]then IDLE_PORT=8081else echo "> 현재 구동중인 포트가 없습니다." echo "> I
DLE_PORT: 8081로 할당" IDLE_PORT=8081fi echo "> $IDLE_PORT 에서 실행중인 도커 컨테이너 종
료"sudo docker stop ${IDLE_PORT}sudo docker rm ${IDLE_PORT}echo "> 도커 이미지 최신 버전 pu
ll"sudo docker pull gkseqls9808/myini-api:latestecho "> 도커 실행 포트:$IDLE_PORT"sudo d
ocker run -d --name $IDLE_PORT -p ${IDLE_PORT}:${IDLE_PORT} -e "server.port=${IDLE_POR
T}" -e TZ=Asia/Seoul gkseqls9808/myini-api:latestecho "> 사용하지 않는 도커 이미지 삭제"doc
ker rmi -f $(docker images -f "dangling=true" -q) || trueecho "> $IDLE_PORT 15 초 후 He
alth Check 시작"echo "> curl -s http://localhost:$IDLE_PORT/actuator/health "sleep 20fo
r RETRY_COUNT in {1..10}do RESPONSE=$(curl -s http://localhost:${IDLE_PORT}/actuator/
health) UP_COUNT=$(echo ${RESPONSE} | grep 'UP' | wc -l) if [ ${UP_COUNT} -ge 1 ] #
$up_count >= 1 ("UP" 문자열이 있는지 검증) then echo "> Health check 성공" echo
"> 전환할 Port: $IDLE_PORT" echo "> Port 전환" echo "set \${service_url} http://1
27.0.0.1:${IDLE_PORT};" | sudo tee /etc/nginx/conf.d/service-url.inc echo "> 엔진엑
스 Reload" sudo service nginx reload break else echo "> Health check의
응답을 알 수 없거나 혹은 실행 상태가 아닙니다." echo "> Health check: ${RESPONSE}" fi i
f [ ${RETRY_COUNT} -eq 10 ] then echo "> Health check 실패. " echo "> 엔진엑스
에 연결하지 않고 배포를 종료합니다." exit 1 fi echo "> Health check 연결 실패. 재시도..." s
leep 10done

```

1. nginx config file 설정

- /etc/nginx/conf.d/service-url.inc

```
set $service_url http://127.0.0.1:8081
```

- /etc/nginx/sites-available/test.conf

```
include /etc/nginx/conf.d/service-url.inc;
```

```

location /api { proxy_pass $service_url; proxy_set_header Host $http_host; pr
oxy_set_header X-Real-IP $remote_addr; proxy_set_header X-Forwarded-For $proxy_add_
x_forwarded_for; proxy_set_header X-Forwarded-Proto $scheme;}

```

4. FrontEnd - 가이드 페이지

4.1. Jenkins 프로젝트 생성

1. 새로운 Item 에서 새 프로젝트를 생성
2. 프로젝트-구성 에서 프로젝트 설정 정보 구성
3. secret token 발급
1~3 과정은 BackEnd와 동일하게 진행

4. nodejs 설정 Dashboard - Jenkins관리 - Global Tool Configuration - NodeJS

버전은 프로젝트 버전과 동일하게 설정한다.

이후 Shell Script를 실행하기 전 Execute NodeJS script 를 통해 nodejs를 먼저 빌드해 준다.

NodeJS

NodeJS installations

List of NodeJS installations on this system

Add NodeJS

NodeJS

Name

node 16.16.0

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 16.16.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Add Installer

Add NodeJS

Execute NodeJS script ?

NodeJS Installation

Specify nodejs installation where npm installed packages to execute the script

node 16.16.0

Script

See [the list of available environment variables](#) accessible by process.env/ENV_VARIABLE.

npmrc file

- use system default -

Cache location

Default (~/npm or %APP_DATA%\npm-cache)

Execute shell ?

5. 빌드 스크립트 Execute Shell

```
cd front-weblsnpm config set proxy nullnpm config set https-proxy nullnpm config set registry http://registry.npmjs.org/npm installnpm run buildcd buildpwdchmod 777 /var/jenkins_home/workspace/front-docs/front-web/buildscp -r -o "StrictHostKeyChecking=no" /v
```

```
ar/jenkins_home/workspace/front-docs/front-web/build ubuntu@k7b203.p.ssafy.io:/home/ub  
untu
```

4.2. GitLab 연동

BackEnd와 동일

4.3. script

- `/etc/nginx/sites-available/test.conf`

```
location / {    root /home/ubuntu/build;    index index.html index.htm;    try_files  
$uri $uri/ /index.html =404;    proxy_set_header Host $http_host;    proxy_set_header  
X-Real-IP $remote_addr;    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_fo  
r;    proxy_set_header X-Forwarded-Proto $scheme;}
```

5. FrontEnd - MyINI App

5.1. 빌드 스크립트

Execute Shell

```
cd frontdocker build -t front-app .docker stop front-appdocker rm front-appdocker run  
--name front-app -d -p 3000:3000 front-app echo "> 사용하지 않는 도커 이미지 삭제"docker rmi  
-f $(docker images -f "dangling=true" -q) || true
```

6. 소셜 로그인을 위한 nginx 설정

```
location /oauth2 {    proxy_pass $service_url;    proxy_set_header Host $http_host;  
proxy_set_header X-Real-IP $remote_addr;    proxy_set_header X-Forwarded-For $proxy_ad  
d_x_forwarded_for;    proxy_set_header X-Scheme $scheme;    proxy_set_header X-Forward  
ed-Proto $scheme;    try_files $uri $uri/ /index.html =404;}location /login/oauth2 {  
proxy_pass $service_url;    proxy_set_header Host $http_host;    proxy_set_header X-Re  
al-IP $remote_addr;    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
proxy_set_header X-Forwarded-Proto $scheme;    try_files $uri $uri/ /index.html =404;}
```

test.conf

```
server {    listen 80;    listen [::]:80;    server_name k7b203.p.ssafy.i  
o;    return 301 https://k7b203.p.ssafy.io$request_uri;}server {    listen 80;  
listen [::]:80;    server_name sub.k7b203.p.ssafy.io;    location / {  
proxy_pass http://127.0.0.1:3000;    }}server {    include /etc/nginx/conf.d/s  
ervice-url.inc;    listen 443 ssl http2;    listen [::]:443;    server_nam
```

```

e k7b203.p.ssafy.io;          # ssl 인증서 적용          ssl_certificate /etc/letsencrypt/liv
e/k7b203.p.ssafy.io/fullchain.pem;          ssl_certificate_key /etc/letsencrypt/live/k7
b203.p.ssafy.io/privkey.pem;          location / {          root /home/ubuntu/buil
d;          index index.html index.htm;          try_files $uri $uri/ /ind
ex.html =404;          proxy_set_header Host $http_host;          proxy_se
t_header X-Real-IP $remote_addr;          proxy_set_header X-Forwarded-For $prox
y_add_x_forwarded_for;          proxy_set_header X-Forwarded-Proto $scheme;
}          location /api {          proxy_pass $service_url;          proxy_
set_header Host $http_host;          proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;          proxy_set_
header X-Forwarded-Proto $scheme;          }          location /oauth2 {          pr
oxy_pass          $service_url;          proxy_set_header H
ost          $http_host;          proxy_set_header X-Real-IP
$remote_addr;          proxy_set_header X-Forwarded-For          $proxy_add_x_for
warded_for;          proxy_set_header X-Scheme          $scheme;
proxy_set_header X-Forwarded-Proto $scheme;          try_files $uri $uri/ /inde
x.html =404;          }          location /login/oauth2 {          proxy_pass
$service_url;          proxy_set_header Host          $http_host;
proxy_set_header X-Real-IP          $remote_addr;          proxy_set_header
X-Forwarded-For          $proxy_add_x_forwarded_for;          proxy_set_header X-
Forwarded-Proto $scheme;          try_files $uri $uri/ /index.html =404;
}}

```

service.conf

```

server {
    listen 80;
    listen [::]:80;

    server_name www.myini.tk myini.tk;

    return https://k7b203.p.ssafy.io$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443;
    server_name www.myini.tk myini.tk;

    # ssl 인증서 적용
    ssl_certificate /etc/letsencrypt/live/www.myini.tk/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/www.myini.tk/privkey.pem;

    location / {
        proxy_pass http://localhost:3000;

        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```


DB

1. ubuntu 패키지 정보 업데이트

```
sudo apt update
```

2. mysql 설치

```
sudo apt install mysql-server
```

3. mysql 설치 확인

```
dpkg -l | grep mysql-server
```

```
root@ip-172-26-3-4:/etc/mysql/mysql.conf.d# dpkg -l | grep mysql-server
ii  mysql-server      8.0.30-0ubuntu0.20.04.2  all          MySQL database server (metapackage depending on the latest version)
ii  mysql-server-8.0  8.0.30-0ubuntu0.20.04.2  amd64        MySQL database server binaries and system database setup
ii  mysql-server-core-8.0  8.0.30-0ubuntu0.20.04.2  amd64        MySQL database server binaries
```

4. mysql 실행여부 확인

```
sudo netstat -tap | grep mysql
```

```
root@ip-172-26-3-4:/etc/mysql/mysql.conf.d# sudo netstat -tap | grep mysql
tcp        0      0 0.0.0.0:33060 0.0.0.0:*        LISTEN      340014/mysqld
tcp        0      0 0.0.0.0:mysql 0.0.0.0:*        LISTEN      340014/mysqld
tcp        0      0 ip-172-26-3-4.ap:mysql 118.42.123.215:63311 ESTABLISHED 340014/mysqld
tcp        0      0 ip-172-26-3-4.ap:mysql 118.42.123.215:63310 ESTABLISHED 340014/mysqld
```

5. mysql 계정설정

```
//접속, 처음 비밀번호는 없으니 엔터
mysql -u root -p
use mysql
select host,user,authentication_string from user;
```

현재 만든 계정 확인

```
mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select Host, User, authentication_string from user;
+-----+-----+-----+
| Host      | User      | authentication_string |
+-----+-----+-----+
| %         | b208      | $A$005$vjh> | 'B%U!zQ|
6qGoaKeruLWl/D/8K41hrTPtiml/09mkQYz0hxdR8LgqK1 |
| %         | root      | *C227B91DF2FFA1D588C82BACFEC17C0014BA675A |
| localhost | debian-sys-maint | $A$005$a {bP\| ::0ND
mCd/NHIZzxPvqbXeDfn4wdGqUudG4Kj5mrWjxkiJLEPCD |
| localhost | mysql.infoschema | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.session | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.sys | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
//root 비밀번호 설정
```

```
alter user 'root'@'localhost' identified with mysql_native_password by '비밀번호 설정';
FLUSH PRIVILEGES;
exit
```

mysql -u root -p 시 이제 비밀번호 치고 들어가야함

6. mysql 외부 접속 설정

```
cd /etc/mysql/mysql.conf.d
sudo nano mysqld.cnf
```

bind-address = 0.0.0.0 으로 수정

```

# The MySQL database server configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user                = mysql
# pid-file           = /var/run/mysqld/mysqld.pid
# socket             = /var/run/mysqld/mysqld.sock
# port               = 3306
# datadir            = /var/lib/mysql

# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir             = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address        = 0.0.0.0
mysqlx-bind-address = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size     = 16M
# max_allowed_packet = 64M
# thread_stack       = 256K

```

[Read 78 lines]

```

//수행
service mysql restart
mysql -u root -p

//원하는 username , password 계정 설정
mysql> create user 'username'@'%' identified by 'password';

//만든계정에 모든 권한 주기 설정
mysql> grant all privileges on . to username@'%';

```