포팅 매뉴얼

📚 Stacks

Management Tool

`Jira` `Mattermost` `Discord` `Notion` `GitLab` `Figma`

IDE

`Visual Studio Code(1.8.6)` `IntelliJ(2023.3.2)`

Infra

`NCP Object Storage` `Nginx(1.18.0)` `Docker(25.0.1)` `Ubuntu(20.04.6)` `EC2`

Frontend

`HTML5` `CSS` `TypeScript(5)` `Next.js(14.1.3)` `React(18)` `React DOM(18)` `Zustand(4.5.2)` `SASS(1.71.1)` `Node.js(20.12.0)` `ESlint(8.57)` `prettier`

Backend

`Java(17)` `SpringBoot(3.2.4)` `JPA` `QueryDSL(5.0.0)` `Redis(7.2.4)` `MySQL(8.3.0)`

Data

`Python(3.9)` `Jupyter notebook` `FastAPI` `Pandas` `Scikit-learn` `SQLAlchemy` `pymysql` `ElasticSearch(8.12.2)` `Kibana(8.12.2)` `Logstash(8.12.2)` `Zookeeper` `Kafka`

🛠️ Infra Settings

ec2 ssh 접속

```
ssh -i k10a601T.pem ubuntu@k10a601.p.ssafy.io
```

Docker 및 Docker Compose 설치

> **Docker 공식 문서 참조**
>
> **https://docs.docker.com/engine/install/ubuntu/**

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gp
g -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/ap
t/keyrings/docker.asc] https://download.docker.com/linux/ub
untu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

```
# docker.sock에 permission denied가 발생할 경우 그룹을 통해 권한
부여
sudo groupadd docker
sudo usermod -aG docker ubuntu
sudo chown root:docker /var/run/docker.sock
```

인프라 구성

> **Docker Compose를 이용해 Nginx, Jenkins, MySQL, Redis를 한번에 구성**

```
vi docker-compose.yml
docker compose up -d
```

```
# docker-compose.yml

version: "3.8"

networks:
  infra:
    driver: bridge

services:
  certbot:
    container_name: certbot
    image: certbot/certbot
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt:rw
      - ./data/certbot/www:/var/www/certbot:rw
    networks:
      - infra
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do ce
rtbot renew; sleep 12h & wait $${!}; done;'"

  nginx:
    container_name: nginx
    image: nginx
    volumes:
      - ./conf/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - 80:80
      - 443:443
    networks:
      - infra

  jenkins:
```

```
    build:
      context: ./jenkins_setup
    container_name: jenkins
    environment:
      - TZ=Asia/Seoul
      - JENKINS_OPTS="--prefix=/jenkins"
    user: root
    privileged: true
    ports:
      - 8000:8080
      - 50000:50000
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - infra

  mysql:
    image: mysql:latest
    container_name: mysql
    expose:
      - "3306"
    environment:
      MYSQL_ROOT_PASSWORD: <password>
      MYSQL_DATABASE: bareun
      MYSQL_USER: <username>
      MYSQL_PASSWORD: <password>
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/config:/etc/mysql/conf.d
      - ./db/mysql/init:/docker-entrypoint-initdb.d
    networks:
      - infra

  redis:
    image: redis:latest
    container_name: redis
```

```
    networks:
      - infra
```

SSL 적용

Let’s Encrypt 를 통해 SSL 인증서 발급

```
mkdir conf
vi conf/nginx.conf
vi init-letsencrypt.sh
sudo ./init-letsencrypt.sh
```

```
# nginx.conf

server {
     listen 80;

     server_name <도메인>;

     location /.well-known/acme-challenge/ {
             allow all;
             root /var/www/certbot;
     }
}
```

```
# init-letsencrypt.sh

#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=<도메인>
```

```
rsa_key_size=4096
data_path="./data/certbot"
email="<이메일>" # Adding a valid address is strongly recomm
ended
staging=0 # Set to 1 if you're testing your setup to avoid
hitting request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and r
eplace existing certificate? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
    exit
  fi
fi


if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ !
-e "$data_path/conf/ssl-dhparams.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/certbo
t/master/certbot-nginx/certbot_nginx/_internal/tls_configs/
options-ssl-nginx.conf > "$data_path/conf/options-ssl-ngin
x.conf"
  curl -s https://raw.githubusercontent.com/certbot/certbo
t/master/certbot/certbot/ssl-dhparams.pem > "$data_path/con
f/ssl-dhparams.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days
1\
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
```

```
    -subj '/CN=localhost'" certbot
echo



echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo



echo "### Requesting Let's Encrypt certificate for $domains
..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
```

```
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

발급된 인증서를 적용하기 위해 nginx.conf 수정

```
# nginx.conf

user nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    client_max_body_size 10M;
    client_body_buffer_size 10M;

    upstream jenkins {
        server jenkins:8080;
    }

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

        server_name <도메인>;
```

```
        location /.well-known/acme-challenge/ {
             allow all;
             root /var/www/certbot;
        }

        location /jenkins {
            return 301 https://$server_name$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name <도메인>;

        ssl_certificate /etc/letsencrypt/live/bareun.life/f
ullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/bareun.li
fe/privkey.pem;

        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location /jenkins {
            proxy_pass         http://jenkins;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }

    log_format  main  '$remote_addr - $remote_user [$time_l
ocal] "$request" '
                      '$status $body_bytes_sent "$http_refe
rer" '
                      '"$http_user_agent" "$http_x_forwarde
```

```
d_for"';
    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;
}
```

Jenkins를 통한 프론트엔드 및 백엔드 CI/CD 구축

> **Jenkins 관리 > Credentail > global > Add Credentials 를 통해 토큰 등록**

```
# ID : IDToken, Name : GitLab API Token

Username : <GitLab ID>
Password : <발급받은 Access Token>
ID : IDToken
```

```
# ID : envFile, Name : .env

NEXT_PUBLIC_SYNCFUSION_KEY="<PUBLIC_SYNCFUSION_KEY>"
NEXT_PUBLIC_BASE_URL="https://<도메인>/api"
NEXT_PUBLIC_OAUTH_KAKAO_URL="https://<도메인>/api/oauth2/aut
horization/kakao"
NEXT_PUBLIC_OAUTH_GOOGLE_URL="https://<도메인>/api/oauth2/au
thorization/google"

NEXT_PUBLIC_API_KEY=AIzaSyAbcSOk4v64N_OtDrAwg6NqCfD4V9ybRVM
NEXT_PUBLIC_MESSAGING_SENDER_ID=440103641572
NEXT_PUBLIC_APP_ID=1:440103641572:web:5da3f683fe540b222827f
1
NEXT_PUBLIC_STORAGE_BUCKET=bareun-life.appspot.com
NEXT_PUBLIC_AUTH_DOMAIN=bareun-life.firebaseapp.com
NEXT_PUBLIC_PROJECT_ID=bareun-life
NEXT_PUBLIC_VAPID_KEY=BIPl9ct1eQFxWBDjPToDKScT3eVOyqXZeuTuP
```

```
0Im4RRZQhiolrGGB1EbErmgtIbYlSwb-990ZXJQeIJeNbIpOGY
NEXT_PUBLIC_ANALYZE=true
```

```
# ID : firebase, Name : firebase-messaging-sw.js

importScripts(
  'https://www.gstatic.com/firebasejs/9.0.2/firebase-app-co
mpat.js',
);
importScripts(
  'https://www.gstatic.com/firebasejs/9.0.2/firebase-messag
ing-compat.js',
);

firebase.initializeApp({
  apiKey: '<api key>',
  authDomain: '<도메인>.firebaseapp.com',
  projectId: '<도메인>',
  storageBucket: '<도메인>.appspot.com',
  messagingSenderId: '440103641572',
  appId: '1:440103641572:web:5da3f683fe540b222827f1',
});

const messaging = firebase.messaging();

self.addEventListener('push', function (e) {
  if (!e.data.json()) return;

  const resultData = e.data.json();

  const notificationTitle = resultData.data.title;
  const notificationOptions = {
    body: resultData.data.body,
    data: resultData.data,
    ...resultData,
  };
  registration.showNotification(notificationTitle, notifica
tionOptions);
```

```
});

self.addEventListener('notificationclick', function (event)
{
  event.notification.close();
  event.waitUntil(clients.openWindow(event.notification.dat
a.url));
});
```

```
# ID : application.yml, Name : application.yml

server:
  port: 8081
  servlet:
    context-path: /api

spring:
  security:
    oauth2:
      client:
        registration:
          google: # registration ID
            client-id: "<google-client-id>"
            client-secret: "<google-client-secret>"
            client-name: GOOGLE
            redirect-uri: "https://<도메인>/api/login/oauth
2/code/google"
            authorization-grant-type: authorization_code
            scope:
              - email
          kakao:
            client-id: "<kakao-client-id>"
            client-secret: "<kakao-client-secret>"
            client-name: KAKAO
            client-authentication-method: client_secret_pos
t
            redirect-uri: "https://<도메인>/api/login/oauth
2/code/kakao"
```

```
            authorization-grant-type: authorization_code
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oaut
h/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/m
e
            user-name-attribute: id
  elasticsearch:
    username: elastic
    password: bareun106
    host: <도메인>
    port: 9200
  data:
    kafka:
      host: 13.209.13.203
      port: 9092
    redis:
      host: redis
      port: 6379

  datasource:
    url: jdbc:mysql://mysql:3306/<db name>?serverTimezone=A
sia/Seoul&useUnicode=true
    username: <username>
    password: <db password>
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: none
    show-sql: false
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100
    defer-datasource-initialization: true
```

```
  sql:
    init:
      mode: always

  servlet:
    multipart:
      maxFileSize: 10MB
      maxRequestSize: 30MB

logging:
  level:
    org:
      springframework:
        security: debug

jwt:
  secret-key: <jwt secret key>
  access-token:
    lifetime: 3600
  refresh-token:
    lifetime: 864000

cloud:
  aws:
    stack:
      auto: false
    region:
      static: kr-standard
    credentials:
      access-key: <access-key>
      secret-key: <secret-key>
    endpoint: https://kr.object.ncloudstorage.com/
    bucket:
      name: bareun-object-storage
    folder:
      tracker: tracker_achieve_image
    default: https://kr.object.ncloudstorage.com//bareun-ob
ject-storage/tracker_achieve_image/fbbf2ce0-86f2-48e9-af0b-
```

```
553e35c6e13e.png

gpt:
  api:
    key: <gpt api key>
    url: https://api.openai.com/v1/chat/completions
    model: gpt-3.5-turbo

fcm:
  certification: bareun-life-firebase-adminsdk-ucver-be590d
c8a0.json
```

```
# ID : firebase, Name : firebase-messaging-sw.js

{
  "type": "service_account",
  "project_id": "<도메인>",
  "private_key_id": "<private_key_id>",
  "private_key": "<private_key>",
  "client_email": "firebase-adminsdk-ucver@<도메인>.iam.gser
viceaccount.com",
  "client_id": "<client_id>",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.co
m/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robo
t/v1/metadata/x509/firebase-adminsdk-ucver%40bareun-life.ia
m.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

Jenkins Pipeline

```
# Server Pipeline
```

```
pipeline {
    agent any

    tools {
        gradle 'Gradle 8.7'
    }

    options {
        disableConcurrentBuilds()
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'dev/BE', credentialsId: 'IDTok
en', url: 'https://lab.ssafy.com/s10-final/S10P31A106'
                dir('backend') {
                    // script {
                    //     sh 'mkdir ./src/main/resources'
                    // }
                    withCredentials([file(credentialsId: 'a
pplication.yml', variable: 'yml')]) {
                        script {
                                sh 'pwd'
                                sh 'cp -f $yml ./src/main/r
esources/application.yml'
                            }
                        }
                    withCredentials([file(credentialsId: 'f
irebase-sdk', variable: 'firebase')]) {
                        script {
                            sh 'cp -f $firebase ./src/main/
resources/bareun-life-firebase-adminsdk-ucver-be590dc8a0.js
on'
                        }
                    }
                }
            }
```

```
        }

        stage('Build') {
            steps {
                dir('backend') {
                    script {
                        sh "chmod +x ./gradlew"
                        sh "./gradlew clean build"
                    }
                }
            }
        }

        stage('Image Build') {
            steps {
                dir('backend') {
                    sh 'docker ps -a -q --filter "name=^/sp
ring-server$" | xargs -r docker stop'
                    sh 'docker ps -a -q --filter "name=^/sp
ring-server$" | xargs -r docker rm'
                    sh 'docker rmi -f spring-server:latest'
                    sh 'docker build -t spring-server .'
                }
            }
        }

        stage('Deploy') {
            steps {
                dir('backend') {
                    sh 'java -version'
                    sh """
                    docker run -d --name spring-server \\
                    --expose 8081 \\
                    --network=ubuntu_infra \\
                    -e TZ=Asia/Seoul \\
                    spring-server
                    """
                }
```

```
            }
        }
    }
}
```

```
# Client-Pipeline

pipeline {
    agent any
    tools {nodejs "node.js"}

    options {
        disableConcurrentBuilds()
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'FE/deploy/S10P31A106-9',
                    credentialsId: 'IDToken',
                    url: 'https://lab.ssafy.com/s10-final/S
10P31A106'
                dir('frontend') {
                    withCredentials([file(credentialsId: 'e
nvFile', variable: 'env')]) {
                        script {
                            sh 'cp -f $env ./.env'
                        }
                    }

                    withCredentials([file(credentialsId: 'f
irebase', variable: 'js')]) {
                        script {
                            sh 'cp -f $js ./public/firebase
-messaging-sw.js'
                        }
                    }
                }
```

```
            }
        }

        stage('Before Build') {
            steps {
                dir('frontend') {
                    sh 'docker ps -a -q --filter "name=^/cl
ient-server$" | xargs -r docker stop'
                    sh 'docker ps -a -q --filter "name=^/cl
ient-server$" | xargs -r docker rm'
                    sh 'docker rmi -f frontend-front:lates
t'
                }
            }
        }

        stage('Deploy') {
            steps {
                dir('frontend') {
                    sh 'docker compose build && docker comp
ose up -d'
                }
            }
        }
    }
}
```

ELK 구축

gitlab clone

```
git clone {gitlab 주소}
```

클론한 디렉토리로 이동 후 .env파일 생성

```
ELASTIC_VERSION=8.12.2

ELASTIC_PASSWORD='bareun106'
LOGSTASH_INTERNAL_PASSWORD='bareun106'
KIBANA_SYSTEM_PASSWORD='bareun106'
METRICBEAT_INTERNAL_PASSWORD=''
FILEBEAT_INTERNAL_PASSWORD=''
HEARTBEAT_INTERNAL_PASSWORD=''
MONITORING_INTERNAL_PASSWORD=''

KAFKA_PORT=0000
KAFKA_SUB_PORT=0000

KIBANA_PORT=0000

LOGSTASH_PORT=0000

ELASTIC_PORT_1=0000
ELASTIC_PORT_2=0000

RANK_LOG=rank_log

ZOOKEEPER_PORT=0000
ZOOKEEPER_CLIENT_PORT=0000
ZOOKEEPER_TICK_TIME=0000

SERVER_IP=서버의 public ip 주소
```

docker compose 실행

```
docker compose build && docker compose up -d
```

만약, kibana에 접속 시 id, password를 적는 화면이 정상적으로 출력되지 않는다면

```
chmod +x {디렉토리}/ELKK/elkk/setup/entrypoin.sh
docker compose up setup
```

추천 서버

Dockerfile 생성

```
mkdir recommend
cd recommend
vi Dockerfile
docker build -t ajisai/weightcloth .
vi docker-compose.yml
docker compose up -d
```

```
# Dockerfile
FROM python:3.11

# 라이브러리 설치
RUN pip install --upgrade pip
RUN pip install --no-cache-dir fastapi
RUN pip install --no-cache-dir uvicorn
RUN pip install --no-cache-dir mysqlclient
RUN pip install --no-cache-dir sqlalchemy
RUN pip install --no-cache-dir scikit-learn
RUN pip install --no-cache-dir pandas
RUN pip install --no-cache-dir openpyxl
RUN pip install --no-cache-dir pyjwt

EXPOSE 9999

WORKDIR /app
CMD ["python", "src/main.py"]
```

```
# docker-compose.yml
```

```
version: "3.8"

networks:
  ubuntu_infra:
    external: true

services:
  recommender:
    image: "ajisai/weightcloth"
    container_name: recommender
    volumes:
      - type: bind
        source: ./app
        target: /app

    networks:
      - ubuntu_infra
```

Nginx Reverse Proxy 적용

> ## nginx.conf를 수정한 뒤에 nginx 재시작

```
vi ./conf/nginx.conf
docker restart nginx
```

```
# nginx.conf

user nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}
```

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    client_max_body_size 10M;
    client_body_buffer_size 10M;

    upstream jenkins {
        server jenkins:8080;
    }

    upstream spring-server {
        server spring-server:8081;
    }

     upstream client-server {
        server client-server:3000;
    }

    upstream recommender {
        server recommender:9999;
    }

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

        server_name <도메인>;

        location /.well-known/acme-challenge/ {
             allow all;
             root /var/www/certbot;
        }

        location /jenkins {
            return 301 https://$server_name$request_uri;
        }
    }
```

```
    server {
        listen 443 ssl;
        server_name <도메인>;

        ssl_certificate /etc/letsencrypt/live/bareun.life/f
ullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/bareun.li
fe/privkey.pem;

        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location /jenkins {
            proxy_pass         http://jenkins;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }

        location /api {
            proxy_pass         http://spring-server;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }


        location /recommendation {
            proxy_pass         http://recommender;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
```

```
            proxy_cache_bypass $http_upgrade;

        }


        location / {
            proxy_pass         http://client-server;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }

    }

    log_format  main  '$remote_addr - $remote_user [$time_l
ocal] "$request" '
                      '$status $body_bytes_sent "$http_refe
rer" '
                      '"$http_user_agent" "$http_x_forwarde
d_for"';
    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;
}
```

⚡ Trouble Shooting

Docker

```
# docker.sock에 permission denied가 발생할 경우 그룹을 통해 권한
부여
sudo groupadd docker
```

```
sudo usermod -aG docker ubuntu
sudo chown root:docker /var/run/docker.sock
```

MySql

```
# MySql 한글 인코딩 문제 발생할 경우
docker exec -it mysql /bin/bash
apt-get update, apt-get install vim
vim /etc/mysql/my.cnf
```

```
# my.cnf

...
[client]
default-character-set=utf8

[mysql]
default-character-set=utf8

[mysqld]
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8
...
```

ELK

```
# 만약 로그를 출력했지만 elasticsearch에 적용되지 않은 경우
1. docker ps -> docker logs {logstash 컨테이너 명}
    - Error 확인, 없다면 다음 단계
2. docker ps -> docker logs {kafka컨테이너 명}
 - Error 확인, 없다면 다음 단계
3. Kibana Port를 확인하고 접속해서 로그가 입력되었는지 확인
    - 보내는 형식이 logstash에서 정상적으로 정제되는지 확인
    - index의 mapping이 의도한대로 이루어져있는지 확인
```