

포팅메뉴얼

- [1. 서버 아키텍처](#)
- [2. 배포 환경/기술 스택](#)
 - [2.1 COMMON](#)
 - [2.2 FRONT](#)
 - [2.3 BACK](#)
 - [2.4 DEVELOP TOOL](#)
- [3. 환경 변수 설정 파일 목록](#)
 - [3.1 Back](#)
 - [3.2 NginX](#)
- [4. 서버 기본 설정하기](#)
 - [4.1 서버 시간 설정](#)
 - [4.2 미러 서버 카카오 서버로 변경](#)
 - [4.3 패키지 목록 업데이트 및 패키지 업데이트](#)
 - [4.4 SWAP 영역 할당](#)
- [5. 방화벽 열기](#)
- [6. 필요한 리소스 설치하기](#)
 - [6.1 Java 설치](#)
 - [6.2 node 설치](#)
 - [6.3 도커 설치](#)
 - [6.3.1 Docker 설치 전 필요한 패키지 설치](#)
 - [6.3.2 Docker에 대한 GPC Key 인증 진행.](#)
 - [6.3.3 Docker 저장소 등록](#)
 - [6.3.3 패키지 리스트 갱신](#)
 - [6.3.4 Docker 패키지 설치](#)
 - [6.3.5 Docker 일반유저에게 권한부여](#)
 - [6.3.6 Docker 서비스 실행](#)
 - [6.3.7 Docker Compose 설치](#)
 - [6.3.8 Docker Compose 실행 권한추가](#)
 - [6.4 MySQL 설치\(in Docker\)](#)
 - [6.4.1 MySQL 실행.](#)
 - [6.4.2 database 생성](#)
 - [6.4.3 MySQL connection 객체 늘려주기](#)
 - [6.6 NginX 설치](#)
- [7. 프로젝트 실행](#)
 - [7.1 프론트](#)
 - [7.1.1 프론트 디렉토리로 이동](#)
 - [7.1.2 npm install, npm run build](#)
 - [7.1.3 nginx 재실행](#)

7.2 백엔드

7.2.1 `application.yml` 입력

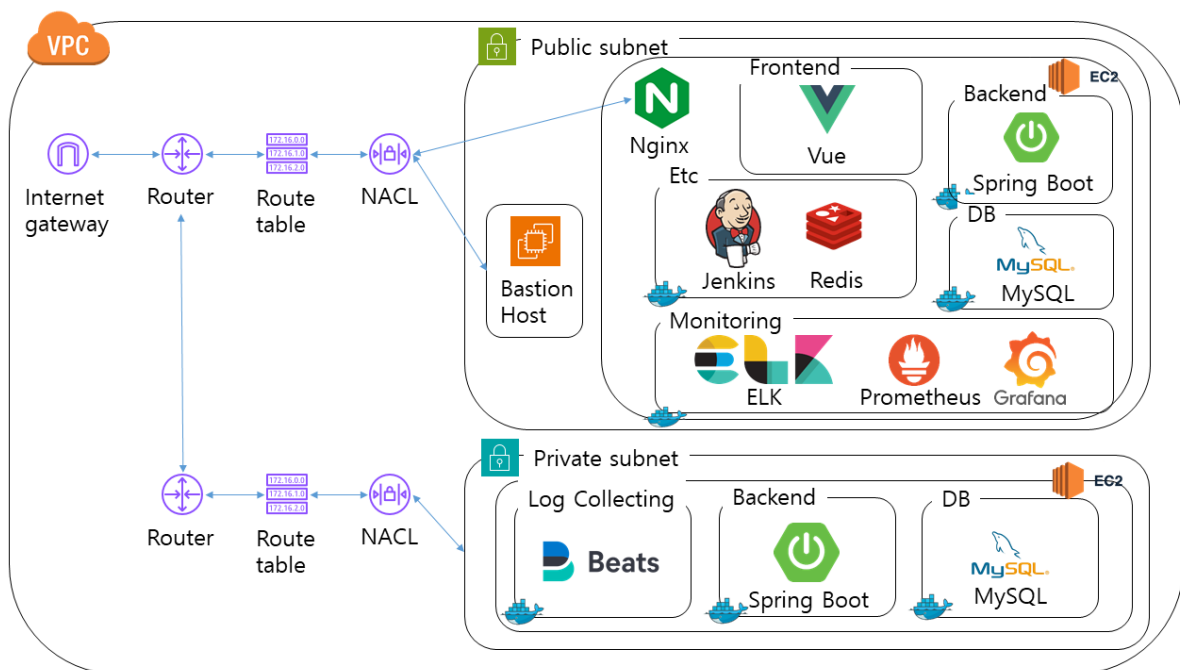
7.2.2 back 디렉터리로 이동

7.2.3 프로젝트 빌드

7.2.4 프로젝트 image 생성

7.2.5 프로젝트 image 실행

1. 서버 아키텍처



2. 배포 환경/기술 스택

2.1 COMMON

- Server : Ubuntu 24.04 LTS
- Docker : 26.1.0
- docker-compose version : 1.29.2
- mysql : 8.3.0
- jenkins : Version 2.440.1(jdk17)

- nginx : 1.24.0

2.2 FRONT

- node : 21.7.3
- npm : 10.5.2
- vue : 3.10.3
- vite : ^5.2.8

2.3 BACK

- jdk : 17
- Spring Boot : 3.2.5
- Gradle : 8.5

2.4 DEVELOP TOOL

- Visual Studio Code
- IntelliJ IDE
- AWS EC2 ㅈ

3. 환경 변수 설정 파일 목록

3.1 Back

- application.yml(소스 폴더)

3.2 NginX

- /etc/nginx/sites-enabled/default

4. 서버 기본 설정하기

서비스는 모든 서버가 하나의 환경 위에서 돌아가도록 설계되었다.

4.1 서버 시간 설정

```
sudo timedatectl set-timezone Asia/Seoul
```

4.2 미리 서버 카카오 서버로 변경

패키지 다운을 빠르게 하기 위함.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g' /etc/apt/sources.list
```

4.3 패키지 목록 업데이트 및 패키지 업데이트

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

4.4 SWAP 영역 할당

```
//용량 확인
```

```
free -h
```

```
//스왑영역 할당
```

```
sudo fallocate -l 20G /swapfile
```

```
//swapfile 관한 수정
```

```
sudo chmod 600 /swapfile
```

```
//swapfile 생성
```

```
sudo mkswap /swapfile
```

```
//swapfile 활성화
```

```
sudo swapon /swapfile
```

```
//시스템 재부팅 되어도 swap 유지할수 있도록 설정.
```

```
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

```
//swap 영역 할당 되었는지 확인  
free -h
```

5. 방화벽 열기

포트번호 : 80 , 443 , 8080 , 8881 , 5044 , 5601 , 9200 , 3000 , 9090 , 6379 , 3306 , 8180

```
sudo ufw allow {포트번호}  
sudo ufw enable  
sudo ufw reload
```

6. 필요한 리소스 설치하기

6.1 Java 설치

```
sudo apt-get install openjdk-17-jdk
```

6.2 node 설치

```
curl -fsSL https://deb.nodesource.com/setup_current.x | sud  
o -E bash -  
sudo apt-get install -y nodejs
```

6.3 도커 설치

6.3.1 Docker 설치 전 필요한 패키지 설치

```
sudo apt-get -y install apt-transport-https ca-certificates  
curl gnupg-agent software-properties-common
```

6.3.2 Docker에 대한 GPG Key 인증 진행.

OK가 떴다면 정상적으로 등록된 것.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

6.3.3 Docker 저장소 등록

```
sudo add-apt-repository "deb [arch=amd64] https://download.do
```

6.3.3 패키지 리스트 갱신

```
sudo apt-get -y update
```

6.3.4 Docker 패키지 설치

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

6.3.5 Docker 일반유저에게 권한부여

```
sudo usermod -aG docker ubuntu
```

6.3.6 Docker 서비스 실행

```
sudo service docker restart
```

6.3.7 Docker Compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

6.3.8 Docker Compose 실행 권한추가

```
sudo chmod +x /usr/local/bin/docker-compose
```

6.4 MySQL 설치(in Docker)

6.4.1 MySQL 실행.

현재 배포된 소스코드는 비밀번호 1234로 맞춰져 있다.

```
docker run -d --restart always -p 3306:3306 -e MYSQL_ROOT_PASSWORD=ssafy605 -v /var/lib/mysql:/var/lib/mysql --name mysql mysql
```

6.4.2 database 생성

- workbench를 이용해서 쿼리문을 실행

```
create database S101;
```

6.4.3 MySQL connection 객체 늘려주기

```
SET GLOBAL max_connections = 300;
```

6.6 NginX 설치

```
sudo apt-get -y install nginx
```

```
sudo snap install --classic certbot
```

```
sudo apt-add-repository -r ppa:certbot/certbot
```

```
sudo apt-get -y install python3-certbot-nginx
```

```
sudo certbot --nginx -d pasds.world  
sudo certbot --nginx -d pasds.world  
sudo certbot --expand -d pasds.world -d pasds.world
```

- /etc/nginx/sites-enabled/default 파일 수정

```

server {

    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
}

server {

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

    root /home/ubuntu/front/dist;

    index index.html index.htm index.nginx-debian.html;
    server_name www.pasds.world pasds.world; # managed by Certbot

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }
}

```



```

        location /app {
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $http_x_for
rwarded_for;
            proxy_pass $service_url;
        }

        location /app/api/notification/stream {
            proxy_pass $service_url;
            proxy_http_version 1.1;
            proxy_set_header Connection "";
            chunked_transfer_encoding off;
            proxy_buffering off;
            proxy_cache off;
        }

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/www.pasds.world/f
ullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/www.pasds.wor
ld/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # mana
ged by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # manage
d by Certbot

}
server {
    if ($host = pasds.world) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

```

```

    if ($host = www.pasds.world) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name www.pasds.world pasds.world;
    return 404; # managed by Certbot

}

```

```
sudo nginx -t
```

```
sudo systemctl restart nginx
```

7. 프로젝트 실행

- 세팅이 끝났다면, 프로젝트를 clone해온다. 첫번째로 GitLab에서 우리 프로젝트를 clone해준다.

```
git clone ${our_git}
```

7.1 프론트

7.1.1 프론트 디렉토리로 이동

```
cd front
```

7.1.2 npm install, npm run build

- front 디렉토리에서 빌드를 진행한다.

```
npm install | npm run build
```

7.1.3 nginx 재실행

```
sudo systemctl restart nginx
```

7.2 백엔드

7.2.1 application. yml 입력

```
server:
  port: 8080

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{mysql주소}:3306/{db이름}
    username: root
    password: {비밀번호}
  elasticsearch:
    uris: http://{elasticsearch주소}
    username: elastic
    password: {비밀번호}
  jpa:
    hibernate:
      ddl-auto: update
  mail:
    host: smtp.gmail.com
    port: 587
    accounts:
      - username: pasdsworld@gmail.com
        password: {비밀번호}
      - username: pasdsworld2@gmail.com
        password: {비밀번호}
      - username: wlsdyd4@gmail.com
        password: {비밀번호}
  properties:
    mail:
```

```

smtp:
  auth: true
  starttls:
    enable: true
    required: true
  connectiontimeout: 5000
  timeout: 5000
  writetimeout: 5000
  auth-code-expiration-millis: 180000 # 3분
data:
  redis:
    host: pasds-redis # 내부 통신망
    # host: pasds.world # public instance
    port: 6379
    password: {비밀번호}
    lettuce:
      pool:
        max-active: 10
        max-idle: 5
        min-idle: 1

security:
  pepper: {비밀번호}
  jwt:
    access-token-expiration-ms: 900000 # 15분
    refresh-token-expiration-ms: 86400000 # 1일
    temporary-token-expiration-ms: 900000 # 15분
    email-token-expiration-ms: 900000 # 15분

# logging:
# level:
# org.springframework.security: DEBUG

cookie:
  path: "/app/api"
  secure: true
  httpOnly: true
  sameSite: "None"

```

```
# sameSite: "Strict" # 마지막 날 설정

kms-server:
  url: http://{kms주소}/kms/api # private instance 내부 통신망

management:
  health:
    mail:
      enabled: false # MailHealthIndicator 비활성화
  endpoints:
    web:
      cors:
        allowed-origins: http://{서버 주소}:9090
      exposure:
        include: "*"
  metrics:
    export:
      prometheus:
        enabled: true
  endpoint:
    health:
      show-details: ALWAYS
```

7.2.2 back 디렉터리로 이동

```
cd back
```

7.2.3 프로젝트 빌드

```
chmod +x ./gradlew
./gradlew wrapper --gradle-version=6.8 --distribution-type=
bin
./gradlew clean bootJar
```

7.2.4 프로젝트 image 생성

```
docker build -t back:latest .
```

7.2.5 프로젝트 image 실행

```
sudo docker run -i -e TZ=Asia/Seoul -e "SPRING_PROFILES_ACT  
IVE=prod" --name back -p 8084:8080 -d back:latest
```