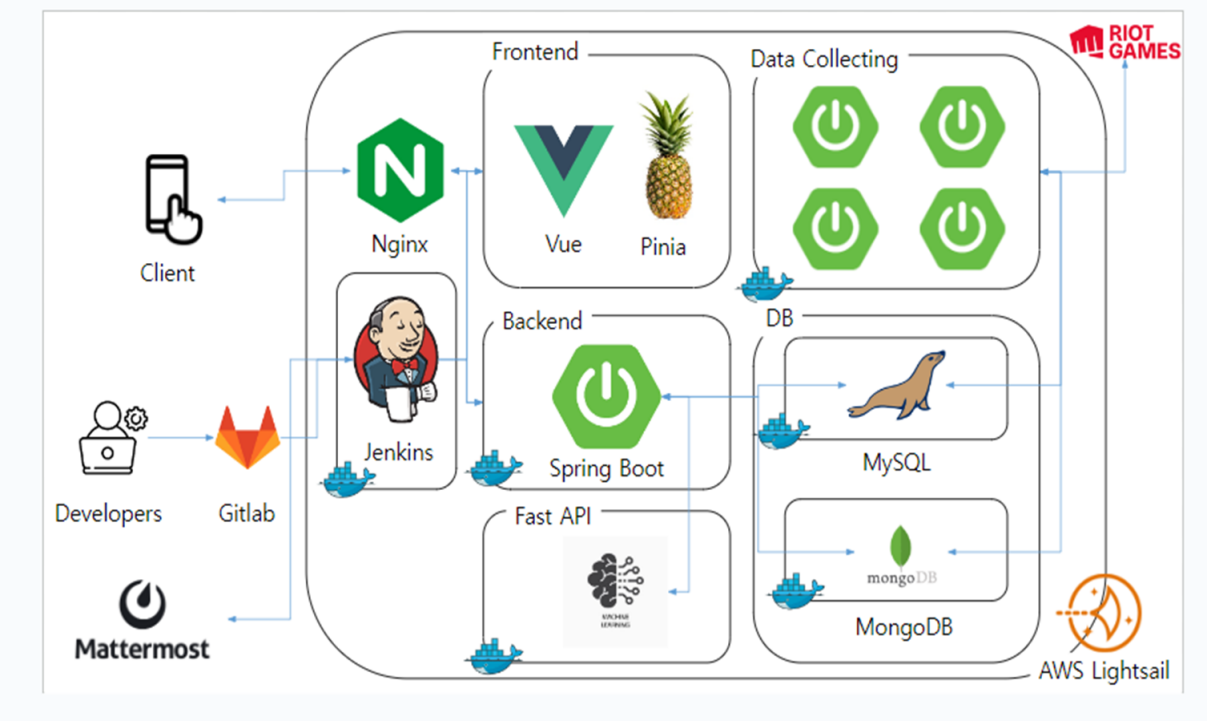
포팅메뉴얼

1. 서버 아키텍처



2. 배포 환경/기술 스택

2.1 COMMON

- Server : Ubuntu 20.04.6 LTS
- Docker : 25.0.4
- docker-compose version : 1.29.2

- mariadb : 8.3.0
- jenkins : Version 2.440.1(jdk17)
- nginx : 1.18.0

2.2 FRONT

- node : v20.11.1
- npm : 10.2.4
- vue : ^3.4.15
- vite : 5.0.11

2.3 BACK

- jdk : 17
- Spring Boot : 3.2.4
- Gradle : 8.5

2.4 DEVELOP TOOL

- Visual Studio Code
- IntelliJ IDE
- AWS EC2
- AWS S3

3. 환경 변수 설정 파일 목록

3.1 Back

- application.properties(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 미러 서버 카카오 서버로 변경

패키지 다운을 빠르게 하기 위함.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirro
r.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 chmode 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/f
stab

//swap 영역 할당 되었는지 확인
free -h
```

5. 방화벽 열기

포트번호 : `80`, `443`, `8080`, `8080`, `8081`, `8084`, `3306`, `27017`

```
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 GPC Key 인증 진행.

OK가 떴다면 정상적으로 등록된 것.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s
udo 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/do
wnload/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_P
ASSWORD=ssafy605 -v /var/lib/mysql:/var/lib/mysql --name my
sql mysql
```

6.4.2 database 생성

- workbench를 이용해서 쿼리문을 실행

```
create database a605;
```

6.4.3 MySQL connection 객체 늘려주기

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

6.5 mongoDB 설치(in Docker)

```
docker run --name mongo -d -p 27017:27017 -v ~/data:/data/d
b mongo
```

```
docker exec -it mongo mongo
```

```
mongosh
```

```
 use admin
```

```
db.createUser({user: "root", pwd: "ssafy605", roles:["roo
t"]})
```

컨테이너삭제후, 볼륨에 저장된 환경으로 run

```
docker run --name mongo -d -p 27017:27017 -v ~/data:/data/d
b mongo --auth
```

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 j10a605.p.ssafy.io
sudo certbot --nginx -d garengg.shop
sudo certbot --expand -d garengg.shop -d www.garengg.shop
```

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

```
# default

upstream backend {
     server 127.0.0.1:8084; # Docker 컨테이너가 호스트의 8084번
포트로 바인딩
}

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

```
        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

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

        location /api {
                proxy_pass http://backend;
        }

}

server {


        root /home/ubuntu/front/dist;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
        server_name j10a605.p.ssafy.io; # managed by Certbo
t

        location /api{
                proxy_pass http://backend;
        }

        location / {
                # First attempt to serve request as file, t
```

```
hen
                # as directory, then fall back to displayin
g a 404.
        #        try_files $uri $uri/ =404;
                try_files $uri $uri/ /index.html; # SPA의 클
라이언트 사이드 라우팅 지원
        }
        location /ws-stomp {
                proxy_pass http://backend;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add
_x_forwarded_for;
                proxy_set_header Host $http_host;

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j10a605.p.ssafy.i
o/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j10a605.p.ssa
fy.io/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 {
        #redirect
    if ($host = j10a605.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
        listen 80 ;
        listen [::]:80 ;
    server_name j10a605.p.ssafy.io;
    return 404; # managed by Certbot


}


server {
        root /home/ubuntu/front/dist;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name garengg.shop www.garengg.shop; # managed by
Certbot

        location / {
                try_files $uri $uri/ /index.html; # SPA의 클
라이언트 사이드 라우팅 지원
        }



        location /api {
                proxy_pass http://backend;
        }

        location /ws-stomp {
                proxy_pass http://backend;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add
_x_forwarded_for;
                proxy_set_header Host $http_host;

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
```

```
        }

    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/garengg.shop/full
chain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/garengg.shop/
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 {
        listen 80 ;
        listen [::]:80 ;
    server_name garengg.shop www.garengg.shop;
return 301 https://$host$request_uri;

}
```

```
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.properties` 입력

```
server:
  port: 8080

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{mysql주소}:3306/{db이름}
    username: root
    password: {비밀번호}
  data:
    mongodb:
      host: {mongodb 주소}
      port: 27017
      authentication-database: admin
      database: {db이름}
      username: root
```

```
      password: {비밀번호}
  jpa:
    hibernate:
      ddl-auto: update
```

7.2.2 `WebSocketConfig.java` 수정

```
...
//14 line
.allowedOrigins(
                        ///CORS 도메인 입력
                )
...
```

7.2.3 back 디렉터리로 이동

```
cd back
```

7.2.4 프로젝트 빌드

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

7.2.5 프로젝트 image 생성

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

7.2.6 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
```

7.3 FAST API

- 패키지 관리를 위해 가상환경 생성 후 라이브러리 설치

```
python --version Python 3.12.1
```

```
python -m venv venv
source venv/Scripts/activate
pip install -r requirements.txt
```

- 라이브 서버 실행

```
cd app
uvicorn main:app --reload
```