



포팅 매뉴얼

☀ 진행상태 진행 중

Project Skill Stack Version

Skill	Version
React	18.3.1
Axios	1.7.7
Vite	5.4.9
TypeScript	5.6.2
eslint	9.13.0
Java	17
SpringBoot	3.3.3
Gradle	8.10.2
MySQL	9.1.0 MySQL Community Server - GPL
Redis	7.4.1
MongoDB	8.0.3
Nginx	1.18.0 (Ubuntu)
Jenkins	2.482
Docker	27.3.1
rabbitMQ	3.13.3

사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost

- 디자인 : Figma
- CI/CD : Jenkins, Docker, DockerHub, Nginx

개발 도구

- Visual Studio Code : 1.90.2
- IntelliJ : IntelliJ IDEA 2024.1.4 (Ultimate Edition)

EC2 포트 번호

Backend : 8080, 8081

채팅 : 5050

Frontend : 5173

MySQL : 3306

Redis : 6379

MongoDB : 27017

Jenkins : 9005

rabbitMQ: 5672, 15672, 61613

CI/CD 구축

백엔드 서버부터 구축

Swap 메모리 설정

여러 빌드 동시 처리 시 물리적 메모리가 가득 찼을 때 추가 작업을 위한 swap 메모리 설정

스왑 메모리 설정

```
// swap 파일을 생성해준다.
```

```
// (메모리 상태 확인 시 swap이 있었지만 디렉토리 파일은 만들어줘야한다.)
```

```
sudo mkdir /var/spool/swap
```

```
sudo touch /var/spool/swap/swapfile
```

```
sudo dd if=/dev/zero of=/var/spool/swap/swapfile count=409600
```

```
// swap 파일을 설정한다.
sudo chmod 600 /var/spool/swap/swapfile
sudo mkswap /var/spool/swap/swapfile
sudo swapon /var/spool/swap/swapfile

// swap 파일을 등록한다.
sudo echo '/var/spool/swap/swapfile none swap defaults 0 0' |

// 메모리 상태 확인
free -h
```

JDK 설치

17로 진행

```
# 업데이트
sudo apt update

# 업그레이드
sudo apt upgrade

# 특정 버전 목록 조회
sudo apt list openjdk-17

# 설치
sudo apt install openjdk-17-jdk

# 설치 확인
java --version
```

Docker 설치

```
# 의존성 설치
sudo apt update
```

```

sudo apt install ca-certificates curl gnupg lsb-release

# 레포지토리
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg

# 레포지토리 추가
echo "deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.gpg] https://download.dock
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | su

# 도커 설치하기
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker

```

Jenkins 설치

Docker outside of Docker (DooD) 방식으로 진행

젠킨스에서 도커 기반의 빌드, 테스트 환경을 직접 관리

```

# 도커 소켓 마운트 하기 (젠킨스 컨테이너에서 도커 명령어 실행되도록 하기)
docker run -itd --name jenkins -p 9005:8080 -v /var/run/docke

# 도커 명령어가 젠킨스에서 실행이 안되거나 권한 오류가 나면 아래 명령어 실행
sudo chmod 666 /var/run/docker.sock

# 젠킨스 컨테이너 비밀번호 확인 명령어
docker exec jenkins cat /var/jenkins_home/secrets/initialAdmi

# 젠킨스 컨테이너로 접속해서 도커 명령어 실행 여부 확인 명령어
docker exec -it <container_name_or_id> /bin/bash
docker exec -it jenkins /bin/bash

# 젠킨스 컨테이너에 접속해서 Docker 명령어 되는지 확인
docker

```

접속 후 테스트

Nginx 설치

```
sudo apt update
sudo apt upgrade
sudo apt install nginx
sudo service nginx start
sudo service nginx status
```

https 설정 (SSL)

무료 Let's Encrypt

```
#Encrypt 설치
sudo apt-get install letsencrypt

#Certbot 설치
sudo apt-get install certbot python3-certbot-nginx

#Certbot 동작 (nginx 중지하고 해야함)
sudo certbot --nginx
1번 방법 sudo certbot --nginx -d [도메인 혹은 ip 주소]
2번 방법 sudo letsencrypt certonly --standalone -d [도메인 혹은 ip 주소]
# 옵션 1번 선택
# 강제 리다이렉트 설정 부분에서 안한다고 함(1번 선택) http 와 https 같이

nginx 설정 적용
sudo service nginx restart
sudo systemctl reload nginx
```

Jenkins, gitLab webhook 설정

깃랩 토큰 발급 → 젠킨스 플러그인 등록(gitlab) → 젠킨스에 API token Credentials 등록
→ 연결확인

Jenkins pipeline 생성

클론을 하기 위한 기본 코드 부터 작성

```
pipeline {
    agent any
    stages {
        stage('Git Clone'){
            steps {
                git branch: 'BE', credentialsId: 'GitLab_Logi
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
    }
}
```

깃랩 웹훅 등록

URL, Secret Token, Trigger 작성

Docker Hub Setting

로그인 후 컨테이너 생성

Docker file 작성

```
# open jdk 17 버전의 환경을 구성
FROM openjdk:17

# tzdata 패키지 설치 및 타임존 설정
RUN ln -snf /usr/share/zoneinfo/Asia/Seoul /etc/localtime &&

# build가 되는 시점에 JAR_FILE이라는 변수 명에 build/libs/*.jar 선언
# build/libs - gradle로 빌드했을 때 jar 파일이 생성되는 경로
ARG JAR_FILE=build/libs/BE-0.0.1-SNAPSHOT.jar

# JAR_FILE을 agaproject.jar로 복사
COPY ${JAR_FILE} ourClass.jar

# 운영 및 개발에서 사용되는 환경 설정을 분리
# -Duser.timezone=Asia/Seoul JVM 옵션을 사용하여 애플리케이션 수준에
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "
```

Jenkins Credential Setting

- JASYPT → Secret text
- Docker Hub → Usernaem with password
- EC2 Server IP → Secret text

SSH 접속설정

plugin 추가(SSh Agent Plugin)

Jenkins Credentials - .pem키 복사붙여넣기

Nginx 설정 변경

무중단 배포 (Blue-Green) 로 진행

무중단 배포 경로를 잡기 위한 service-url.inc, Deploy File 따로 작성

Frontend는 기본 포트

Backend는 /api 밑으로 들어오면 8080,8081로 연결(socket통신 포함)

nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

worker_rlimit_nofile 2048;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

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

    ##
```



```

# SSL Settings
##

ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3
ssl_prefer_server_ciphers on;

##
# Logging Settings
##

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;

##
# Gzip Settings
##

gzip on;

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
# gzip_types text/plain text/css application/json application/javascript;

##
# Virtual Host Configs
##

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}

#mail {
#    # See sample authentication script at:
#    # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript

```

```
#
# # auth_http localhost/auth.php;
# # pop3_capabilities "TOP" "USER";
# # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
# server {
#     listen    localhost:110;
#     protocol  pop3;
#     proxy     on;
# }
#
# server {
#     listen    localhost:143;
#     protocol  imap;
#     proxy     on;
# }
#}
```

nginx/sites-enabled/default

```
server {
    listen 80;
    listen [::]:80 ;
    server_name meongspot.kro.kr;

    location / {
        return 301 https://$host$request_uri;
    }
}

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

    listen 443 ssl;
```

```

listen [::]:443 ssl;

ssl_certificate /etc/letsencrypt/live/meongspot.kro.k
ssl_certificate_key /etc/letsencrypt/live/meongspot.k
include /etc/letsencrypt/options-ssl-nginx.conf; # ma
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # mana

server_name meongspot.kro.kr;


root /var/www/html;
index index.html index.htm index.nginx-debian.html;

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


location / {
    root /home/ubuntu/Frontend/dist;
    index index.html;
    try_files $uri $uri/ /index.html;
}


location /api/ {
    proxy_pass http://localhost:8
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_read_timeout 20m;

    add_header 'Access-Control-Allow-Origin' '*';
    #add_header 'Access-Control-Allow-Methods' 'G
    add_header 'Access-Control-Allow-Headers' 'Or

```

```

        add_header 'Access-Control-Allow-Credentials'

        if ($request_method = 'OPTIONS') {
            return 204;
        }

    }

}

```

service_url.inc

```

# service_url 주소 할당
set $service_url http://127.0.0.1:8081;

```

Jenkins pipeline BE 작성

```

pipeline {
    agent any

    tools {
        jdk("jdk17")
    }

    environment {
        JAVA_HOME = 'C:\\Program Files\\Java\\jdk-17'
        PATH = "${JAVA_HOME}\\bin;${env.PATH}"
    }

    stages {
        stage('Git Clone') {
            steps {

```

```

        git branch: 'back/develop', credentialsId: 'G
    }
    post {
        failure {
            echo 'Repository clone failure !'
        }
        success {
            echo 'Repository clone success !'
        }
    }
}
stage('secret.yml copy') {
    steps {
        withCredentials([file(credentialsId: 'applica
            script {
                sh 'docker cp $secretFile jenkins:/va
            }
        }
    }
}
stage('Build') {
    steps {
        // 프로젝트 권한 변경
        sh 'chmod +x ./BE/gradlew'
        // 프로젝트 빌드
        sh 'cd ./BE && ./gradlew clean build -x test'
    }
}
stage('Docker Hub Login') {
    steps {
        withCredentials([usernamePassword(credentials
            sh 'echo "$DOCKER_PASSWORD" | docker logi
        }
    }
}
stage('Docker Build and Push') {
    steps {
        withCredentials([usernamePassword(credentialsI

```

```

        dir('./BE') {
            // JAR 파일 위치 확인
            sh 'ls -la build/libs/'

            // Docker 빌드
            sh 'docker build -f Dockerfile -t $DOCKER_IMAGE_NAME'

            // Docker 푸시
            sh 'docker push $DOCKER_REPO/$DOCKER_IMAGE_NAME'

            echo 'Docker push Success!!'
        }
    }
}

stage('Deploy') {
    steps {
        sshagent(credentials: ['my-ssh-credentials'])
        withCredentials([string(credentialsId: 'E', variable: 'SSH_KEY')]) {
            sh 'ssh -o StrictHostKeyChecking=no user@host:command'
        }
    }
}

// stage('Notification') {
//     steps {
//         echo 'jenkins notification!'
//     }
//     post {
//         success {
//             script {
//                 def Author_ID = sh(script: "git show --pretty=format: %H", returnStdout: true).trim()
//                 def Author_Name = sh(script: "git show --pretty=format: %an", returnStdout: true).trim()
//                 mattermostSend(
//                     color: 'good',
//                     message: "빌드 성공: ${env.JOB_NAME} ${Author_Name} ${Author_ID}",
//                     endpoint: 'https://meeting.ssa.com/api/v1/messages',
//                     channel: 'd208_build_result'
//                 )
//             }
//         }
//     }
// }

```

```

//          )
//      }
//  }
//      failure {
//          script {
//              def Author_ID = sh(script: "git sh
//              def Author_Name = sh(script: "git
//              mattermostSend(
//                  color: 'danger',
//                  message: "빌드 실패: ${env.JOB_N
//                  endpoint: 'https://meeting.ssa
//                  channel: 'd208_build_result'
//              )
//          }
//      }
//  }
// }
}
}

```

Deploy File 작성

```

# Docker Hub 로그인 추가
#echo "$DOCKER_PASSWORD" | sudo docker login -u "$DOCKER_USER

# 0
# 이미지 갱신
sudo docker compose -p meongspot-8080 -f /home/ubuntu/docker-
sudo docker compose -p meongspot-8081 -f /home/ubuntu/docker-

#1
EXIST_GITCHAN=$(sudo docker compose -p meongspot-8080 -f dock

if [ -z "$EXIST_GITCHAN" ]; then
    echo "8080 컨테이너 실행"
    sudo docker compose -p meongspot-8080 -f /home/ubuntu

```

```

        BEFORE_COLOR="8081"
        AFTER_COLOR="8080"
        BEFORE_PORT=8081
        AFTER_PORT=8080
    else
        echo "8081 컨테이너 실행"
        sudo docker compose -p meongspot-8081 -f /home/ubuntu
        BEFORE_COLOR="8080"
        AFTER_COLOR="8081"
        BEFORE_PORT=8080
        AFTER_PORT=8081
    fi

    echo "${AFTER_COLOR} server up(port:${AFTER_PORT})"

# 2
# for cnt in `seq 1 10`;
# do
#     echo "서버 응답 확인하는중~(${cnt}/10)";
#     UP=$(curl -s http://127.0.0.1:${AFTER_PORT}/api/health-
#     if [ "${UP}" != "OK" ]; then
#         sleep 10
#         continue
#     else
#         break
#     fi
# done

# if [ $cnt -eq 10 ]; then
#     echo "서버에 문제가 있어요..."
#     exit 1
# fi

# 3
#sudo sed -i "s/${BEFORE_PORT}/${AFTER_PORT}/" /etc/nginx/conf
#sudo nginx -s reload
#echo "Deploy Completed!!"

```



```

# Nginx 설정 파일 직접 수정
NginxConfig="/etc/nginx/sites-enabled/default"
if [ -f "$NginxConfig" ]; then
    # 현재 포트를 찾고 새로운 포트로 변경
    sudo cp "$NginxConfig" "$NginxConfig.bak" # 백업

    sudo sed -i "s|proxy_pass http://localhost:[0-9]*;/|proxy_pas

# 변경된 내용을 확인
if grep -q "proxy_pass http://localhost:${AFTER_PORT}/;" "$Ng
    echo "Nginx 설정 파일 업데이트 성공"
else
    echo "Nginx 설정 파일 업데이트 실패"
    exit 1
fi
else
    echo "Nginx 설정 파일이 존재하지 않습니다."
    exit 1
fi

# Nginx 설정 테스트 및 재시작
if sudo nginx -t && sudo systemctl reload nginx; then
    echo "Nginx 재시작 성공"
else
    echo "Nginx 재시작 실패"
    exit 1
fi

# Nginx 재시작
sudo nginx -s reload
echo "Deploy Completed!!"

# 4
echo "$BEFORE_COLOR server down(port:${BEFORE_PORT})"
sudo docker compose -p meongspot-${BEFORE_COLOR} -f docker-co

# 5
sudo docker image prune -f

```

Docker-compose File 작성

docker-compose.meongspot8080.yml

```
version: '3.1'

services:
  api:
    image: llovehate/meongspot:latest
    container_name: meongspot-8080
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8080
      # - jasypt.encryptor.key=[KEY VALUE]
    ports:
      - '8080:8080'
```

docker-compose.meongspot8081.yml

```
version: '3.1'

services:
  api:
    image: llovehate/meongspot:latest
    container_name: meongspot-8081
    environment:
      - TZ=Asia/Seoul
      - LANG=ko_KR.UTF-8
      - HTTP_PORT=8081
      # - jasypt.encryptor.key=[KEY VALUE]
    ports:
      - '8081:8080'
```

CHAT 서버 환경 구축

deploy_chat.sh 작성

```
# Docker Hub 로그인 추가
# echo "$DOCKER_PASSWORD" | sudo docker login -u "$DOCKER_USERNAME" -p "$DOCKER_PASSWORD"

# 0
# 최신 이미지 가져오기
sudo docker compose -p meongspot -f /home/ubuntu/docker-compose.yml pull

# 1
# 현재 실행 중인 컨테이너가 있으면 중지
EXIST_CONTAINER=$(sudo docker compose -p meongspot -f /home/ubuntu/docker-compose.yml ps --filter status=running --format '{{.Name}}')

if [ -n "$EXIST_CONTAINER" ]; then
    echo "기존 컨테이너 중지"
    sudo docker compose -p meongspot -f /home/ubuntu/docker-compose.yml stop
fi

# 2
# 새로운 컨테이너 실행
echo "새 컨테이너 실행"
sudo docker compose -p meongspot -f /home/ubuntu/docker-compose.yml up --build
echo "컨테이너가 성공적으로 시작되었습니다."

# 3
# Nginx 설정 파일에서 포트 변경 필요 없이 기본 설정 유지

# 4
# Nginx 설정 테스트 및 재시작
if sudo nginx -t && sudo systemctl reload nginx; then
    echo "Nginx 재시작 성공"
else
    echo "Nginx 재시작 실패"
    exit 1
fi
```

```
# 5
# 사용하지 않는 이미지 제거
sudo docker image prune -f

echo "Deploy Completed!!"
```

Jenkins pipeline CHAT 작성

```
pipeline {
    agent any

    tools {
        jdk("jdk17")
    }

    environment {
        JAVA_HOME = 'C:\\Program Files\\Java\\jdk-17'
        PATH = "${JAVA_HOME}\\bin;${env.PATH}"
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'back/develop', credentialsId: 'G
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        stage('secret.yml copy') {
```

```

        steps {
            withCredentials([file(credentialsId: 'application_credentials', variable: 'secretFile')]) {
                script {
                    sh 'docker cp $secretFile jenkins:/var/run/docker.sock'
                }
            }
        }
    }
}

stage('Build') {
    steps {
        // 프로젝트 권한 변경
        sh 'chmod +x ./CHAT/gradlew'
        // 프로젝트 빌드
        sh 'cd ./CHAT && ./gradlew clean build -x test'
    }
}

stage('Docker Hub Login') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'docker_credentials', usernameVariable: 'DOCKER_USERNAME', passwordVariable: 'DOCKER_PASSWORD')]) {
            sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
        }
    }
}

stage('Docker Build and Push') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'docker_credentials', usernameVariable: 'DOCKER_USERNAME', passwordVariable: 'DOCKER_PASSWORD')]) {
            dir('./CHAT') {
                // JAR 파일 위치 확인
                sh 'ls -la build/libs/'

                // Docker 빌드
                sh 'docker build -f Dockerfile -t $DOCKER_REPO/$DOCKER_IMAGE .'

                // Docker 푸시
                sh 'docker push $DOCKER_REPO/$DOCKER_IMAGE'

                echo 'Docker push Success!!'
            }
        }
    }
}

```



```

if sudo nginx -t && sudo systemctl reload nginx; then
    echo "Nginx 재시작 성공"
else
    echo "Nginx 재시작 실패"
    exit 1
fi

sudo docker image prune -f

echo "Deploy Completed!!"

```

Jenkins pipeline GPS 작성

```

pipeline {
    agent any

    tools {
        jdk("jdk17")
    }

    environment {
        JAVA_HOME = 'C:\\Program Files\\Java\\jdk-17'
        PATH = "${JAVA_HOME}\\bin;${env.PATH}"
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'back/develop', credentialsId: 'G
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
    }
}

```

```

    }
  }
}
stage('secret.yml copy') {
  steps {
    withCredentials([file(credentialsId: 'applica
      script {
        sh 'docker cp $secretFile jenkins:/va
      }
    }
  }
}
stage('Build') {
  steps {
    // 프로젝트 권한 변경
    sh 'chmod +x ./gps/gradlew'
    // 프로젝트 빌드
    sh 'cd ./gps && ./gradlew clean build -x test
  }
}
stage('Docker Hub Login') {
  steps {
    withCredentials([usernamePassword(credentials
      sh 'echo "$DOCKER_PASSWORD" | docker logi
    }
  }
}
stage('Docker Build and Push') {
  steps {
    withCredentials([usernamePassword(credentialsId
      dir('./gps') {
        // JAR 파일 위치 확인
        sh 'ls -la build/libs/'

        // Docker 빌드
        sh 'docker build -f Dockerfile -t $DOCK

        // Docker 푸시

```



```

        sh 'docker push $DOCKER_REPO/$DOCKER_PR
    }
    echo 'Docker push Success!!'
}
}
}
stage('Deploy') {
    steps {
        sshagent(credentials: ['my-ssh-credentials'])
            withCredentials([string(credentialsId: 'E
                sh 'ssh -o StrictHostKeyChecking=no u
            }
        }
    }
}
}
```

프론트 엔드 환경 구축

Jenkins Plugins에 NodeJS 추가

Google Maps Platform | Google for Developers

수백만 개의 웹사이트와 앱이 Google Maps Platform을 사용하여
사용자에게 효과적인 서비스 환경을 제공하고 있습니다.

 <https://developers.google.com/maps?hl=ko>



Jenkins pipeline 작성

위 백엔드 파이프라인에 통합 구현

deploy_fe.sh 작성

```
#!/bin/bash

# dist 디렉토리의 경로를 정의
DIST_DIR="/home/ubuntu/Frontend/dist"

# 압축 해제할 임시 디렉토리 생성
TEMP_DIR="/home/ubuntu/Frontend/temp"
sudo mkdir -p "$TEMP_DIR"

# 압축 해제
tar -xvf /home/ubuntu/Frontend/front_0.1.0.tar -C "$TEMP_DIR"

# 기존 dist 디렉토리가 있다면 삭제
if [ -d "$DIST_DIR" ]; then
    sudo rm -rf "$DIST_DIR"
fi

# dist 디렉토리를 생성
sudo mkdir -p "$DIST_DIR"

# 임시 디렉토리의 내용을 dist로 이동
sudo mv "$TEMP_DIR"/* "$DIST_DIR"

# 임시 디렉토리 삭제
sudo rm -rf "$TEMP_DIR"

# Nginx 설정 재적용
sudo nginx -s reload
```

Jenkins pipeline FE 작성

```
pipeline {
    agent any
```

```

stages {
  stage('Git Clone'){
    steps {
      git branch: 'front/develop', credentialsId: '
    }
    post {
      failure {
        echo 'Repository clone failure !'
      }
      success {
        echo 'Repository clone success !'
      }
    }
  }

  stage('Cleanup') {
    steps {
      sh 'rm -f ./frontend/front_0.1.0.tar'
      echo 'Cleanup success !'
    }
  }

  stage('Build') {
    steps {
      dir("./FE") {
        nodejs(nodeJSInstallationName: 'NodeJS 20
        sh 'rm -rf node_modules'
        sh 'rm -rf package-lock.json'
        sh 'CI=false yarn install'
        sh 'CI=false yarn build'
      }
    }
    echo 'FE Build success !'
  }

  stage('Compression') {

```



```
VITE_KAKAO_MAP_API_K
VITE_FIREBASE_API_KEY
VITE_FIREBASE_AUTH_DOMAIN
VITE_FIREBASE_PROJECT_ID
VITE_FIREBASE_STORAGE_BUCKET
VITE_FIREBASE_MESSAGING_SENDER_ID
VITE_FIREBASE_APP_ID
VITE_FIREBASE_VAPID_KEY
```

application-secret.yml

```
spring:
  jwt:
    secret: d107neverforgetkeyyourprojectismeongspotthatcanmake
  datasource:
    url: jdbc:mysql://meongspot.kro.kr:3306/meongspot?useSSL=
#    url: jdbc:mysql://localhost:3306/meongspotlocal?useSSL=f
  data:
    redis:
      host: meongspot.kro.kr
      port: 6379

    mongodb:
      host: meongspot.kro.kr
      port: 27017
      database: meongspot
      authentication-database: admin

sms:
  api:
    key:
    secret:
    url: https://api.coolsms.co.kr

cloud:
```

```
aws:
  s3:
    bucket: meongspotd107
  credentials:
    access-key:
    secret-key:
  region:
    static: ap-northeast-2
    auto: false
  stack:
    auto: false

fcm :
  firebase_config_path : meongspot-firebase-adminsdk-qrwnh-17
```

시연 시나리오

시연

두명이 진행(한명은 자리에 앉아서 파트너)

- 반려견 등록
- 지도기반 모임 확인, 추천 모임 확인
- 모임 참여(강아지 정보 확인)
- 모임 참여자들이랑 채팅하기
- 산책 시작, 종료 (알림 확인)
- 산책 기록 확인
- 친구 검색해서 친구 추가
- 친구랑 채팅하기

1. 팀원 한명이 지금 산책을 하러 나가있는데, 산책 시작을 해보겠다.
2. 산책 시작 후 실시간 로그 찍히는 것까지 확인
3. 그럼 산책하고 있는 동안 모임 기능을 살펴보겠다.

4. 이 지역 재검색으로 내 주변 모임을 확인할 수 있네요 ~~~
5. 내 근처에 모임을 한 번 만들어볼게요 ~~~ (진평동 강아지 모여라, 아무나
6. 주변에서 핫한 모임이 무엇인지 확인해볼까요?
7. 오 ~~~ 내 주변에서 모임이 가장 많은 곳을 추천해주는 멍스팟 추천을 보니
8. 동락공원에 있는 모임 하나 살펴보기
9. 이런 강아지들이 있고, 조금 더 자세하게 보고 싶으면 사용자 클릭 → 반려견
10. 저희 강아지와 성향이 잘 맞을 것 같은데요? 한번 가입해볼게요
11. 가입 후 채팅
12. 이 모임에 있는 '웅'이라는 사용자와 친해지고 싶은데, 친구 신청을 보내!
13. 오 ~~ 마이페이지 내부에서 친구 목록을 확인할 수 있네요
14. 모임 기능을 다 살펴봤다. 마침 산책이 끝났다고 하는데, 산책 기록이 어떻
15. 산책 로그 확인

외부 서비스

- 카카오맵 API