



포팅 메뉴얼

개발 환경 세팅

Frontend

Backend

Server

Docker, Docker-compose 설치

- Docker

```
//패키지 업데이트
sudo apt update

//도커 다운로드를 위한 https 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-common

//도커 레포지토리 접근을 위한 GPG Key 설정
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

//도커 레포지토리 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

//패키지 업데이트
sudo apt update

//도커 설치
sudo apt install docker-ce

//도커 실행 상태 확인
sudo systemctl status docker
```

- Docker-compose

```
//도커 컴포즈 설치
sudo curl -L "https://github.com/docker/compose/releases/download/v2.5.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

//도커 컴포즈 권한 부여
sudo chmod +x /usr/local/bin/docker-compose

//심볼릭 링크 연결
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

//도커 컴포즈 버전 확인
docker-compose --version
```

NGINX 설치 / 세팅

https 담당 외부 NGINX

- nginx.conf

```
worker_processes auto; # 워커 프로세스 수

events {
    worker_connections 1024; # 최대 연결 수
}

http {
    server {
        listen 443 ssl;
        listen [::]:443 ssl;

        # SSL 공개키, 개인키 경로 명시

        ssl_certificate /etc/ssl/cert.pem;
        ssl_certificate_key /etc/ssl/key.pem;

        # SSL 암호화 프로토콜 설정
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;

        # React 앱을 위한 location 블록 (8888 포트로 프록시)
        location / {
            proxy_pass http://j11a604.p.ssafy.io:8888; # React 앱이 실행 중인
            proxy_set_header Host $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;

            # CORS 설정 추가
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT,
            add_header 'Access-Control-Allow-Headers' 'Origin, Authorization, Ac
            add_header 'Access-Control-Allow-Credentials' 'true'; # 자격 증명

            # OPTIONS 요청에 대한 응답 (프리플라이트 요청 처리)
            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
                add_header 'Access-Control-Allow-Headers' 'Origin, Authorization
                add_header 'Access-Control-Allow-Credentials' 'true';
                return 204; # No Content 상태로 응답
            }
        }

        # API 요청을 위한 location 블록 (80 포트로 프록시)
        location /api/ {
            proxy_pass http://j11a604.p.ssafy.io:80; # API 서버가 실행 중인 서
            proxy_set_header Host $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;

            # CORS 설정 추가
            add_header 'Access-Control-Allow-Origin' '*';
```

```

        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT,
        add_header 'Access-Control-Allow-Headers' 'Origin, Authorization, Ac
        add_header 'Access-Control-Allow-Credentials' 'true'; # 자격 증명

# OPTIONS 요청에 대한 응답 (프리플라이트 요청 처리)
if ($request_method = 'OPTIONS') {
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
    add_header 'Access-Control-Allow-Headers' 'Origin, Authorization
    add_header 'Access-Control-Allow-Credentials' 'true';
    return 204; # No Content 상태로 응답
}
}

error_page 500 502 503 504 /50x.html;

location = /50x.html {
    root /usr/share/nginx/html;
}
}
}

```

- docker-compose.yml

```

version: "3.7"
services:
  nginx:
    container_name: nginx
    image: nginx:latest
    ports:
      - "443:443"
    volumes:
      - /home/ubuntu/nginx/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/live/j11a604.p.ssafy.io-0003/fullchain.pem:/etc/ssl/cert.pem
      - /etc/letsencrypt/live/j11a604.p.ssafy.io-0003/privkey.pem:/etc/ssl/key.pem
    restart: always

```

웹 서버 담당 NGINX

- nginx.conf

```

events {
    worker_connections 1024; # 동시에 처리할 수 있는 최대 연결 수
}

http {

    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;

    types {
        application/javascript js;
    }
    server {
        listen 8888; # 서버가 수신할 포트 번호

```

```

server_name  j11a604.p.ssafy.io;  # 서버 이름 설정

location / {
    root      /usr/share/nginx/html;  # 정적 파일이 위치한 루트 디렉토리
    index     index.html index.html;  # 기본 인덱스 파일
    try_files $uri $uri/ /index.html;  # 요청한 파일이 없을 경우 index.html로 포워딩 (리액트)

    # CORS 설정 추가
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE';
    add_header 'Access-Control-Allow-Headers' 'Origin, Authorization, Accept, Content-Type';
    add_header 'Access-Control-Allow-Credentials' 'true';  # 자격 증명 허용 (예: 쿠키, 인토큰)

    # OPTIONS 요청에 대한 응답 (프리플라이트 요청 처리)
    if ($request_method = 'OPTIONS') {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE';
        add_header 'Access-Control-Allow-Headers' 'Origin, Authorization, Accept, Content-Type';
        add_header 'Access-Control-Allow-Credentials' 'true';
        return 204;  # No Content 상태로 응답
    }
}

# 에러 페이지 설정 (예시)
error_page 404 /404.html;  # 404 에러 발생 시 보여줄 페이지
location = /404.html {
    internal;  # 내부 요청으로만 접근 가능
}
}
}

```

Jenkins 설치 / 실행

- docker-compose.yml

```

version: '3.8'

services:
  jenkins:
    image: jenkins/jenkins:lts
    restart: unless-stopped
    container_name: jenkins
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"          # Docker 소켓 공유
      - "/home/ubuntu/jenkins-data:/var/jenkins_home"        # Jenkins 데이터 저장
      - "./jenkins-docker-install.sh:/jenkins-docker-install.sh
      - "/etc/letsencrypt/archive/j11a604.p.ssafy.io-0003:/ssl/keys"
    environment:
      - TZ=Asia/Seoul  # 타임존 설정 추가
    user: "root"
    networks:
      - my_network  # my_network에 연결

networks:

```

```
my_network: # 이 부분은 네트워크를 정의하지 않고 사용만 합니다.
external: true # 기존 네트워크를 사용하도록 설정
```

- Jenkins 컨테이너에서 도커를 사용하기 위한 sh 파일 : jenkins-docker-install.sh

```
apt-get update
apt-get install ca-certificates curl gnupg

install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/debian \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
tee /etc/apt/sources.list.d/docker.list > /dev/null

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

- Jenkins 접속 주소

```
http://j11a604.p.ssafy.io:8080
```

- Pipeline 설정

```
pipeline {
    agent any

    stages {
        stage('Repository clone') {
            steps {
                sh 'pwd'
                git branch: 'notification', credentialsId: 'qkrtprjs', url: 'https://lab.ssafy.io:8080'
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        stage('Build image') {
            steps {
                dir('notification') {
                    sh 'chmod +x ./gradlew'
                    // 환경 변수 설정
                    sh '''
                        export DB_HOST=j11a604.p.ssafy.io
                        export DB_PORT=3306
                        export DB_USER=root
                        export DB_PASSWORD=ssafy
                    '''
                }
            }
        }
    }
}
```

```

        export DB_NAME=trabean

        ./gradlew build
    ...
    sh 'pwd'
    sh '''
        docker build -t qkrtprjs/notification \
        --build-arg DB_HOST=j11a604.p.ssafy.io \
        --build-arg DB_PORT=3306 \
        --build-arg DB_USER=root \
        --build-arg DB_PASSWORD=ssafy \
        --build-arg DB_NAME=trabean .
    ...
    }
    echo 'Build image...'
}
post {
    failure {
        echo 'Build image failure !'
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'danger',
            message: "도커 이미지 빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
            endpoint: 'https://meeting.ssafy.com/hooks/bb6j17ansjnambc9cjddf8gw7c
            channel: 'CICD'
            )
        }
    }
    success {
        echo 'Build image success !'
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
            mattermostSend (color: 'good',
            message: "도커 이미지 빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
            endpoint: 'https://meeting.ssafy.com/hooks/bb6j17ansjnambc9cjddf8gw7c
            channel: 'CICD'
            )
        }
    }
}
}

stage('Remove Previous image') {
    steps {
        script {
            try {
                sh 'docker stop notification'
                sh 'docker rm notification'
            } catch (e) {
                echo 'fail to stop and remove container'
            }
        }
    }
}
post {
    failure {
        echo 'Remove Previous image failure !'
    }
}

```

```

    }
    success {
        echo 'Remove Previous image success !'
    }
}
stage('Run New image') {
    steps {
        sh '''
            docker run --name notification -d \
            -p 8083:8083 \
            -e DB_HOST=j11a604.p.ssafy.io \
            -e DB_PORT=3306 \
            -e DB_USER=root \
            -e DB_PASSWORD=ssafy \
            -e DB_NAME=trabean \
            -e TZ=Asia/Seoul \
            qkrtprrjs/notification
        '''
        echo 'Run New member image'
    }
    post {
        failure {
            echo 'Run New image failure !'
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true)
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true)
                mattermostSend (color: 'danger',
                    message: "서비스 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID} ${Author_Name}",
                    endpoint: 'https://meeting.ssafy.com/hooks/bb6j17ansjnambc9cjddf8gw7c',
                    channel: 'CICD'
                )
            }
        }
        success {
            echo 'Run New image success !'
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true)
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true)
                mattermostSend (color: 'good',
                    message: "서비스 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID} ${Author_Name}",
                    endpoint: 'https://meeting.ssafy.com/hooks/bb6j17ansjnambc9cjddf8gw7c',
                    channel: 'CICD'
                )
            }
        }
    }
}
}
```

- 브랜치에 접근할 수 있도록 Credentials 설정

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	qkrtpjrs	qkrtpjrs456@naver.com/***** (qkrtpjrs)
		System	(global)	ProjectAccessToken	GitLab API token (ProjectAccessToken)
		System	(global)	aws_key	aws_key
		System	(global)	gitLabProjectAPIToken2	GitLab API token (gitLabProjectAPIToken2)

- ProjectAccessToken : Git lab 프로젝트에 접근할 수 있도록 설정
- qkrtpjrs : Git lab 계정
- Tool 설정

NodeJS

Name

NodeJS 20.18.0

☒ Install automatically

?

Install from nodejs.org

Version

NodeJS 20.18.0

For the underlying architecture, if available, force the installation of the 32bit package. C
 ☐ 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 syntax `packageName@version`

yarn@1.22.21

Global npm packages refresh hours

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

72

Add Installer ▼

- GitLab 웹훅 설정

Webhooks		
Project hooks 0/10 ADD NEW WEBHOOK		
http://j11a604.p.ssafy.io:8080/project/notification Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/user Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/account Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/payment Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/travel Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/chatbot Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/gateway Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/eureka Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/frontend Push events SSL Verification: enabled	Test ▾	Edit Delete
http://j11a604.p.ssafy.io:8080/project/frontendTest Push events SSL Verification: enabled	Test ▾	Edit Delete

- 각각의 브랜치별 웹훅 설정

- Jenkins - GitLab 연결 시크릿 토큰

Server Port Number

80 : Gateway

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/gateway>

secret token : a4d32daa89df5a1b3934f8034b0d54a8

8761 : EurekaServer

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/eureka>

secret token : e1300ecf0fcf62dd1b37f585af470c38

8080 : Jenkins

8081 : AccountServer

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/account>

secret token : 621708e5c12ba1f9fb58cfe810d712c6

8082 : chatbot

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/chatbot>

secret token : ce3990243bbcb9477879203350f20421

8083 : notification

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/notification>

secret token : b155b8b8879499b5f05d2745eb8f875c

8084 : payment

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/payment>

secret token : 916489341839219ba03f7b6aa619f6bd

8085 : travel

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/travel>

secret token : 6647027bbe6fc73ec94c5a5f8f023bcb

8086 : user

webhook URL :

<http://j11a604.p.ssafy.io:8080/project/user>

secret token : d52ccbf3eecb478f6de746014319c8ab

3306 : MySQLServer

8888 : frontend
webhook URL :
<http://j11a604.p.ssafy.io:8080/project/frontend>
secret token : 198ef89c08245bba7c12610d44f12f41

CI/CD를 위한 Dockerfile

- Backend

```
# Eclipse Temurin 17 JDK 이미지를 기반으로 합니다.
FROM eclipse-temurin:17-jdk

# 빌드 시 사용할 ARG 변수를 정의합니다.
ARG DB_HOST
ARG DB_PORT
ARG DB_USER
ARG DB_PASSWORD
ARG DB_NAME

# 작업 디렉토리를 /app으로 설정합니다.
WORKDIR /app

# 빌드된 JAR 파일을 /app 디렉토리에 복사합니다.
COPY build/libs/notification-0.0.1-SNAPSHOT.jar app.jar

# ARG 변수를 ENV로 설정하여 컨테이너 실행 시 사용할 수 있도록 합니다.
ENV DB_HOST=${DB_HOST}
ENV DB_PORT=${DB_PORT}
ENV DB_USER=${DB_USER}
ENV DB_PASSWORD=${DB_PASSWORD}
ENV DB_NAME=${DB_NAME}

# 컨테이너가 사용하는 포트 8083을 노출합니다.
EXPOSE 8083

# 컨테이너가 시작될 때 실행할 명령을 설정합니다.
ENTRYPOINT ["java", "-jar", "app.jar"]

# 컨테이너에 /data 디렉토리를 볼륨으로 설정합니다.
VOLUME ["/data"]
```

- Frontend

```
# 빌드 단계 설정
FROM node:20 AS build-stage
# 작업 디렉토리를 /app으로 설정
WORKDIR /app
# package.json과 package-lock.json 파일을 현재 작업 디렉토리로 복사
COPY package*.json ./

# Yarn을 사용하여 의존성 설치
RUN npm ci
# 모든 소스 파일을 현재 작업 디렉토리로 복사
COPY . .
# ARG 추가
ARG REACT_APP_END_POINT
ENV REACT_APP_END_POINT=$REACT_APP_END_POINT
# 애플리케이션 빌드
```

```

RUN npm run build
#RUN echo "REACT_APP_END_POINT: $REACT_APP_END_POINT" && yarn build

# 프로덕션 단계 설정
FROM nginx:alpine AS production-stage
# 빌드 단계에서 생성된 빌드 결과물을 Nginx의 기본 HTML 디렉토리로 복사
COPY --from=build-stage /app/build /usr/share/nginx/html

# 커스텀 Nginx 설정 파일을 Nginx의 설정 디렉토리로 복사
COPY nginx.conf /etc/nginx/nginx.conf

# Nginx가 사용할 포트 80을 노출
EXPOSE 8888
# Nginx를 포그라운드 모드로 실행하여 컨테이너가 계속 실행되도록 함
CMD ["nginx", "-g", "daemon off;"]

```

- Spring Cloud Gateway

```

# Eclipse Temurin 17 JDK 이미지를 기반으로 합니다.
FROM eclipse-temurin:17-jdk

# 작업 디렉토리를 /app으로 설정합니다.
WORKDIR /app

# 빌드된 JAR 파일을 /app 디렉토리에 복사합니다.
COPY build/libs/gateway-0.0.1-SNAPSHOT.jar app.jar

# 컨테이너가 사용하는 포트 80 노출합니다.
EXPOSE 80

# 컨테이너가 시작될 때 실행할 명령을 설정합니다.
ENTRYPOINT ["java", "-jar", "app.jar"]

# 컨테이너에 /data 디렉토리를 볼륨으로 설정합니다.
VOLUME ["/data"]

```

- Eureka Server

```

# Eclipse Temurin 17 JDK 이미지를 기반으로 합니다.
FROM eclipse-temurin:17-jdk

# 작업 디렉토리를 /app으로 설정합니다.
WORKDIR /app

# 빌드된 JAR 파일을 /app 디렉토리에 복사합니다.
COPY build/libs/eureka-0.0.1-SNAPSHOT.jar app.jar

# 컨테이너가 사용하는 포트 80 노출합니다.
EXPOSE 8761

# 컨테이너가 시작될 때 실행할 명령을 설정합니다.
ENTRYPOINT ["java", "-jar", "app.jar"]

```

```
# 컨테이너에 /data 디렉토리를 볼륨으로 설정합니다.
VOLUME ["/data"]
```

환경 변수

```
db:
  DB_HOST=j11a604.p.ssafy.io
  DB_PORT=3306
  DB_USER=root
  DB_PASSWORD=ssafy
  DB_NAME=trabean
SSAFY_API:
  API_KEY=1b5cd29adccc46609ff1ce0a589584e0
  PEPPER=ac052c71c986b28977452cb7ffa50f78725c7f5a52264d639561866f31edf26e
frontend:
  REACT_APP_END_POINT=https://j11a604.p.ssafy.io
```

사용 포트

Jenkins	8080
Nginx(외부)	443
Nginx(내부)	8888
Gateway	80
Eureka	8761
Account(MicroServer)	8081
Chatbot(MicroServer)	8082
Notification(MicroServer)	8083
Payment(MicroServer)	8084
Travel(MicroServer)	8085
User(MicroServer)	8086
MySQL	3306