📜

포팅 매뉴얼

💡 **배포 순서**

1. be, fe에 도커파일 생성 (Git에 존재)
2. nginx, certbot 컨테이너 실행
3. certbot으로 ssl 인증서 발급
4. 정상 발급되면 나머지 컨테이너도 띄움
5. nginx conf에서 / -> 3000, /api -> 8081 로 리버스 프록시
6. 젠킨스 관리자 계정 생성, 필요한 플러그인 설치
7. 젠킨스 설정에서 gitlab - credentials - GitLab API token add 하고 깃랩에서 생성한 토큰 넣어줌
8. 젠킨스 파이프라인 프로젝트 생성
9. 프로젝트 설정에서 빌드 트리거 - Build when a change is push to GitLab 체크
10. 프로젝트 설정에서 빌드 트리거 - 고급에서 시크릿 토큰 생성
11. 깃랩 webhook 설정으로 가서 젠킨스 프로젝트 설정에 있는 웹훅 url이랑 시크릿 토큰 넣고 트리거 될 브랜치명 작성함
12. 젠킨스 Global Tool Configuration 설정에서 jdk, gradle, nodejs 설정해줌
    12-1. 서버에서 젠킨스 컨테이너에 들어감
    12-2. openjdk11 설치
    12-3. env 치면 나오는 JAVA_HOME을 젠킨스 G.T.C 설정에 적어줌 (자동 설치는 8까지 밖에 안돼서)
13. 파이프라인 작성
    13-1. 빌드 후 도커 허브에 푸시하고 서버에 ssh 접속하여 docker-compose 명령어 실행
    13-2. 깃 풀 해오는 step에서 credentialsId 넣을 때 위에서 만든 GitLab API token ID는 인식이 안돼서 크레덴셜에 username&password 도 하나 추가함

Docker

`/docker-compose.yml`

```
version: '3'
services:
  nginx:
    container_name: nginx
    image: nginx:latest
    restart: always
    volumes:
      - ./data/nginx/conf.d:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - 80:80
      - 443:443
    command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"

  certbot:
    container_name: certbot
    image: certbot/certbot
    tty: true
    restart: always
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"

  mariadb:
    container_name: mariadb
    image: mariadb:latest
    restart: always
    volumes:
      - ./data/mariadb/conf.d:/etc/mysql/conf.d
      - ./data/mariadb/data:/var/lib/mysql
    env_file: ./data/mariadb/.env
    environment:
      TZ: Asia/Seoul
    networks:
      - backend
    ports:
      - 3306:3306

  fe:
    container_name: fe
    image: jiyoonbyeon/moalarm-fe
    restart: always
    ports:
      - 3000:80

  msa-gateway:
    container_name: msa-gateway
    image: jiyoonbyeon/moalarm-msa-gateway
    restart: always
    networks:
      - backend
    ports:
      - 8081:8080

  msa-auth:
    container_name: msa-auth
    image: jiyoonbyeon/moalarm-msa-auth
    restart: always
    networks:
      - backend
    ports:
      - 8082:8080

  msa-member:
    container_name: msa-member
    image: jiyoonbyeon/moalarm-msa-member
    restart: always
    networks:
      - backend
    ports:
      - 8083:8080

  msa-alarm:
    container_name: msa-alarm
    image: jiyoonbyeon/moalarm-msa-alarm
    restart: always
    networks:
      - backend
    ports:
      - 8084:8080

  msa-history:
    container_name: msa-history
    image: jiyoonbyeon/moalarm-msa-history
    restart: always
    networks:
      - backend
    ports:
      - 8085:8080

  jenkins:
    build:
      context: ./data/jenkins
    container_name: jenkins
    image: jenkins/latest
    user: root
    privileged: true
    restart: always
    volumes:
      - ./data/jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 8080:8080
      - 50000:50000


networks:
  backend:
```

NGINX

`/data/nginx/conf.d/app.conf`

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

     server_name {도메인 이름} {도메인 이름 2};

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

     location / {
        return 308 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name {도메인 이름} {도메인 이름 2};
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/{도메인 이름}/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/{도메인 이름}/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://{도메인 이름}:3000;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP           $remote_addr;
        proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
    }

    location /api {
        proxy_pass http://{도메인 이름}:8081;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Real-IP           $remote_addr;
        proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
    }
}
```

JENKINS

`/data/jenkins/Dockerfile`

```
FROM jenkins/jenkins:lts-jdk11

USER root
```

```
# install docker
RUN apt-get update && \
    apt-get -y install apt-transport-https \
        ca-certificates \
        curl \
        gnupg2 \
        zip \
        unzip \
        software-properties-common && \
    curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
    add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
    $(lsb_release -cs) \
    stable" && \
    apt-get update && \
    apt-get -y install docker-ce
```

`Pipeline`

[moalarm-fe]

```
pipeline {
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s08-final/S08P31A407.git"
    }

    tools {
        gradle 'gradle-7.6.1'
    }

    stages {
        stage('Pull Git Branch') {
            steps {
                git url: "${GIT_URL}", branch: "front/develop", credentialsId: 'GITLAB_AUTH', poll: true, changelog: true
            }
            post {
                failure {
                  echo 'Pull Git Branch failure !'
                }
                success {
                  echo 'Pull Git Branch success !'
                }
          }
        }

        stage('FE Docker Build') {
            steps {
                dir('./FE/moalarm') {
                    sh 'docker build -t jiyoonbyeon/moalarm-fe .'
                }
            }
            post {
                failure {
                  echo 'Docker build failure !'
                }
                success {
                  echo 'Docker build success !'
                }
            }
        }

        stage('FE Docker Push') {
            steps {
                sh 'docker push jiyoonbyeon/moalarm-fe'
            }
            post {
                failure {
                  echo 'Docker push failure !'
                }
                success {
                  echo 'Docker push success !'
                }
            }
        }
        stage('FE Deploy') {
            steps {
                sshagent (credentials: ['EC2']) {
                    sh """
                    ssh -o StrictHostKeyChecking=no ubuntu@{도메인 이름} '
                    sudo docker-compose up -d --build fe
                    '
                    """
                }
            }
            post {
                failure {
                  echo 'Deploy failure !'
                }
                success {
                  echo 'Deploy success !'
                }
            }
        }
    }
}
```

[moalarm-msa-alarm]

(gateway/auth/member/history도 브랜치, 이미지 명 등 제외하고 동일함)

```
pipeline {
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s08-final/S08P31A407.git"
    }

    tools {
        gradle 'gradle-7.6.1'
    }

    stages {
        stage('Pull Git Branch') {
            steps {
                git url: "${GIT_URL}", branch: "msa/alarm/develop", credentialsId: 'GITLAB_AUTH', poll: true, changelog: true
            }
            post {
                failure {
                  echo 'Pull Git Branch failure !'
                }
                success {
                  echo 'Pull Git Branch success !'
                }
          }
        }

        stage('BE Build') {
            steps {
                dir('./alarm') {
                    sh 'cp -r /var/jenkins_home/alarm/resources ./src/main'
                    sh 'chmod +x gradlew'
                    sh 'gradle wrap'
                    sh './gradlew clean bootJar'
                }
            }
            post {
                failure {
                  echo 'Gradle jar build failure !'
                }
                success {
                  echo 'Gradle jar build success !'
                }
            }
        }

        stage('BE Docker Build') {
            steps {
                dir('./alarm') {
                    sh 'docker build -t jiyoonbyeon/moalarm-msa-alarm .'
                }
            }
            post {
                failure {
                  echo 'Docker build failure !'
                }
                success {
                  echo 'Docker build success !'
                }
            }
        }

        stage('BE Docker Push') {
            steps {
                sh 'docker push jiyoonbyeon/moalarm-msa-alarm'
            }
            post {
                failure {
                  echo 'Docker push failure !'
                }
                success {
                  echo 'Docker push success !'
                }
            }
        }

        stage('BE Deploy') {
            steps {
                sshagent (credentials: ['EC2']) {
                    sh """
                    ssh -o StrictHostKeyChecking=no ubuntu@{도메인 이름} '
                    sudo docker-compose up -d --build msa-alarm
                    '
                    """
                }
            }
            post {
                failure {
                  echo 'Deploy failure !'
                }
                success {
                  echo 'Deploy success !'
                }
            }
        }
    }
}
```

MariaDB

`/data/mariadb/.env`

```
MYSQL_HOST=localhost
MYSQL_PORT=3306
MYSQL_ROOT_PASSWORD={root 비밀번호}
MYSQL_DATABASE=moalarm
MYSQL_USER={유저명}
MYSQL_PASSWORD={비밀번호}
```

Sping boot

- **gateway**

  `/data/jenkins/gateway/resources/application-dev.yml`

  ```
  spring:
    cloud:
      gateway:
        default-filters:
  ```

```
          - DedupeResponseHeader=Access-Control-Allow-Origin Access-Control-Allow-Credentials
        routes:
          - id: auth_route
            uri: http://k8a407.p.ssafy.io:8082
            predicates:
              - Path=/api/v2/auth/**
          - id: channel_route
            uri: http://k8a407.p.ssafy.io:8083
            predicates:
              - Path=/api/v2/channels/**
            filters:
              - JwtDecodeFilter
          - id: key_route
            uri: http://k8a407.p.ssafy.io:8083
            predicates:
              - Path=/api/v2/key/**
            filters:
              - JwtDecodeFilter
          - id: member_route
            uri: http://k8a407.p.ssafy.io:8083
            predicates:
              - Path=/api/v2/member/**
            filters:
              - JwtDecodeFilter
          - id: hist_route
            uri: http://k8a407.p.ssafy.io:8085
            predicates:
              - Path=/api/v2/history/**
            filters:
              - JwtDecodeFilter
          - id: alarm_route
            uri: http://k8a407.p.ssafy.io:8084
            predicates:
              - Path=/api/v2/notification/**


jwt:
  secret: {jwt 시크릿 값}
  expire-day: 30

security:
  allowed-origins:
    http://localhost:5500,
    http://127.0.0.1:5500,
    https://k8a407.p.ssafy.io,
    https://moalarm600.com

server:
  max-http-header-size: 16384

crypto:
  secret: {시크릿 값}
  salt: {salt 값}

logging:
  level:
    root: trace
```

- **auth**

  `/data/jenkins/auth/resources/application-dev.yml`

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    username: {유저명}
    password: {비밀번호}
    url: jdbc:mariadb://mariadb:3306/moalarm

  jpa:
    generate-ddl: true

crypto:
  secret: {시크릿 값}
  salt: {salt 값}
```

```
jwt:
  secret: {jwt 시크릿 값}
  expire-day: 30
```

- **member**

  `/data/jenkins/member/resources/application-dev.yml`

```
server:
  port: 8080
  servlet:
    context-path: /api/v2

spring:
  datasource:
    driverClassName: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://mariadb:3306/moalarm
    username: {유저명}
    password: {비밀번호}
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: update

  jackson:
    default-property-inclusion: non_null

crypto:
  secret: {시크릿 값}
  salt: {salt 값}

security:
  allowed-origins:
    http://localhost:5500,
    http://127.0.0.1:5500,
    https://k8a407.p.ssafy.io,
    https://moalarm600.com
```

- **alarm**

  `/data/jenkins/alarm/resources/application-dev.yml`

```
mail:
  smtp:
    auth: true
    starttls:
      required: true
      enable: true
    socketFactory:
      class: javax.net.ssl.SSLSocketFactory
      fallback: false
      port: 465
    ssl:
      checkServerIdentity: true

url:
  member: http://k8a407.p.ssafy.io:8083/api/v2/channels/secret
  history: http://k8a407.p.ssafy.io:8085/api/v2/history
  alarmRequest: http://k8a407.p.ssafy.io:8085/api/v2/history/alarmRequest
```

- **history**

  `/data/jenkins/history/resources/application-dev.yml`

```
spring:
  datasource:
    driverClassName: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://mariadb:3306/moalarm
    username: {유저명}
    password: {비밀번호}
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true

crypto:
  secret: {시크릿 값}
  salt: {salt 값}

security:
  allowed-origins:
    http://localhost:5500,
    http://127.0.0.1:5500,
    https://k8a407.p.ssafy.io,
    https://moalarm600.com
```