빌드&배포 매뉴얼 및 외부 서비스 정보

[빌드&배포 활용 매뉴얼]

- JVM: OpenJDK 17 JVM
- 웹 서버: Nginx
  - **[설정파일]**
    - /etc/nginx/nginx.conf

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

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;
        client_max_body_size 100M;
        # 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 SSLv
        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 applicatio

        ##
        # Virtual Host Configs
        ##

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

- /etc/nginx/sites-avaliable/default

```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name i11c109.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/i11c109.p.ssafy.io/fullcha
    ssl_certificate_key /etc/letsencrypt/live/i11c109.p.ssafy.io/pri
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed b
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by

        location ~ ^/(api|oauth2|login)/ {
            proxy_pass http://localhost:8090;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
            proxy_set_header X-Forwarded-Proto $scheme;

            proxy_buffer_size          128k;
            proxy_buffers              4 256k;
```

```
            proxy_busy_buffers_size    256k;


        }

        location /ws-stomp {  # WebSocket 경로
            proxy_pass http://localhost:8090;  # 백엔드 서버 주소
            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_forwarded_
            proxy_set_header X-Forwarded-Proto $scheme;

            # 타임아웃 설정 (필요시 조정)
            proxy_read_timeout 60s;
            proxy_send_timeout 60s;
        }

        location / {
            proxy_pass http://localhost:5173;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
            proxy_set_header X-Forwarded-Proto $scheme;

            proxy_buffer_size          128k;
            proxy_buffers              4 256k;
            proxy_busy_buffers_size    256k;
        }
}
```

- Jenkinsfile

```
pipeline {
    agent any

    environment {
        GITLAB_REPOSITORY_URL = credentials('GITLAB_REPOSITORY_URL')
        DOCKERHUB_USERNAME = credentials('DOCKERHUB_USERNAME')
        DOCKERHUB_PASSWORD = credentials('DOCKERHUB_PASSWORD')
        DOCKERHUB_REPOSITORY = credentials('DOCKERHUB_REPOSITORY')
        EC2_INSTANCE_PRIVATE_KEY = credentials('EC2_INSTANCE_PRIVATE_KEY')
        EC2_INSTANCE_PORT = 22
        DOCKERHUB_NAME = 'leadme'
        VM_OPTION_NAME = credentials('VM_OPTION_NAME')
        VM_OPTION_PASSWORD = credentials('VM_OPTION_PASSWORD')
```

```
        MONGO_USERNAME = credentials('MONGO_USERNAME') // 사용하지 않음
        MONGO_PASSWORD = credentials('MONGO_PASSWORD') // 사용하지 않음
    }

    stages {
        stage('Clone Repository') {
            steps {
                script {
                    sh 'rm -rf S11P12C109'
                    echo "Cloning repository from: ${GITLAB_REPOSITORY_URL
                    sh "git clone ${GITLAB_REPOSITORY_URL}"
                }
            }
        }

        stage('Build Project') {
            steps {
                script {
                    dir('S11P12C109/server') {
                        sh 'chmod +x ./gradlew'

                        sh './gradlew clean build -x test'
                    }

                    // 클라이언트 빌드
                    dir('S11P12C109/client') {
                        sh 'npm install'
                        sh 'npm run build'
                    }
                }
            }
        }

        stage('Docker Build and Push Java Docker Image') {
            steps {
                script {
                    dir('S11P12C109/server') {
                        withCredentials([usernamePassword(credentialsId: '
                            sh '''
                            echo $DOCKERHUB_PASSWORD | docker login -u $DO
                            docker build -t ${DOCKERHUB_USERNAME}/${DOCKER
                            docker push ${DOCKERHUB_USERNAME}/${DOCKERHUB_
                            '''
                        }
                    }
                }
            }
        }
```

```
        stage('Build and Push Client Docker Image') {
            steps {
                script {
                    dir('S11P12C109/client') { // 클라이언트 디렉토리 경로 변경
                        withCredentials([usernamePassword(credentialsId: '
                            sh '''
                            echo $DOCKERHUB_PASSWORD | docker login -u $DO
                            docker build -t ${DOCKERHUB_USERNAME}/client-i
                            docker push ${DOCKERHUB_USERNAME}/client-image
                            '''
                        }
                    }
                }
            }
        }

        stage('Build and Push Python Docker Image') {
            steps {
                script {
                    dir('S11P12C109/leadme') {
                        sh 'docker stop python-container || true'
                        sh 'docker rm python-container || true'

                        withCredentials([usernamePassword(credentialsId: '
                            sh '''
                            echo $DOCKERHUB_PASSWORD | docker login -u $DO
                            '''

                            sh '''
                            docker build -t ${DOCKERHUB_USERNAME}/python-i
                            docker push ${DOCKERHUB_USERNAME}/python-image
                            '''
                        }

                        // 컨테이너 실행
                        sh '''
                            docker run -d \
                            --name python-container \
                            -p 4567:4567 \
                            -v /home/ubuntu/leadme/video/temporary:/home/u
                            -v /home/ubuntu/leadme/video/user:/home/ubuntu
                            -v /home/ubuntu/leadme/video/challenge:/home/u
                            -v /home/ubuntu/leadme/video/challenge/audio:/
                            -v /home/ubuntu/leadme/video/temporary/thumbna
                            -v /home/ubuntu/leadme/video/user/thumbnail:/h
                            ${DOCKERHUB_USERNAME}/python-image:latest
                        '''
```

```
                }
            }
        }
    }



    stage('Deploy to EC2') {
        steps {
            script {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'ubuntu',
                            transfers: [
                                sshTransfer(
                                    sourceFiles: '',
                                    execCommand: """
                                    docker pull ${DOCKERHUB_USERNAME}/
                                    docker stop ${DOCKERHUB_NAME} || t
                                    docker rm ${DOCKERHUB_NAME} || tru
                                    docker run --name ${DOCKERHUB_NAME
                                -v /home/ubuntu/leadme/video/temporary:/ho
                                -v /home/ubuntu/leadme/video/user:/home/ub
                                -v /home/ubuntu/leadme/video/challenge:/ho
                                -v /home/ubuntu/leadme/video/challenge/aud
                                -v /home/ubuntu/leadme/video/temporary/thu
                                -v /home/ubuntu/leadme/video/user/thumbnai
                                -v /home/ubuntu:/host \
                                -e JAVA_OPTS="-D${VM_OPTION_NAME}=${VM_OPT
                                ${DOCKERHUB_USERNAME}/${DOCKERHUB_REPOSITO

                                    docker pull ${DOCKERHUB_USERNAME}/
                                    docker stop client || true
                                    docker rm client || true
                                    docker run --name client -d -p 517

                                    docker image prune -f
                                    """,
                                    execTimeout: 120000
                                )
                            ],
                            usePromotionTimestamp: false,
                            alwaysPublishFromMaster: false,
                            retry: 1,
                            verbose: true
                        )
```

```
                        ]
                    )
                }
            }
        }
    }

    post {
        always {
            echo 'Pipeline completed.'
        }
        success {
            echo 'Build was successful!'
        }
        failure {
            echo 'Build failed.'
        }
    }
}
```

- Jenkins Credential
  - GITLAB_REPOSITORY_URL=`https://lab.ssafy.com/s11-webmobile1-sub2/S11P12C109`
  - DOCKERHUB_USERNAME=`imnunu`
  - DOCKERHUB_PASSWORD=(개인 도커허브 비밀번호를 넣었습니다.)
  - DOCKERHUB_REPOSITORY=`leadme`
  - EC2_INSTANCE_PRIVATE_KEY=(.pem 파일을 넣었습니다.)
  - VM_OPTION_NAME=`crypto.password`
  - VM_OPTION_PASSWORD=`leadmessafy11`

- IDE: **IntelliJ IDEA**(Spring Boot), **Visual Studio Code**(React.js)
- DB접속 관련 프로퍼티

```
spring:
    datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://i11c109.p.ssafy.io:3306/leadme?useSSL=false&serverTi
        username: ssafy
    password: ssafy

  data:
    mongodb:
      uri: mongodb://leadme:leadmessafy11@i11c109.p.ssafy.io:27070/local?a
    redis:
      host: i11c109.p.ssafy.io
```

```
      port: 6380
      password: leadmeredis109

jwt:
  issuer: secretkey@gmail.com
  secret_key: leadme

python-server:
  url:  http://i11c109.p.ssafy.io:4567
  temp-directory: /home/ubuntu/python/video/temporary
  permanent-directory: /home/ubuntu/python/video/user
  permanent-challenge-directory: /home/ubuntu/python/video/challenge
  temporary-thumbnail-directory: /home/ubuntu/python/video/temporary/thumb
  permanent-thumbnail-directory: /home/ubuntu/python/video/user/thumbnail
  youtube-audio-directory: /home/ubuntu/python/video/challenge/audio

openvidu:
  url: https://i11c109.p.ssafy.io:8443
  secret: leadme
```

- 배포 특이사항
  - /client/Dockerfile

```
FROM node:20

WORKDIR /

COPY . .

RUN npm i

RUN npm run build

RUN apt-get update && apt-get install nginx -y

# 로컬에서 빌드한 결과물을 /var/www/html 으로 복사합니다.
COPY ./dist /var/www/html

RUN rm /etc/nginx/sites-enabled/default
COPY nginx/default /etc/nginx/sites-available/default
RUN ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/d

# 컨테이너의 80번 포트를 열어줍니다.
EXPOSE 80

# nginx 서버를 실행하고 백그라운드로 동작하도록 합니다.
CMD ["nginx", "-g", "daemon off;"]
```

        - 초기에는 docker image를 run시킬 때 index.html 정적 파일을 ec2 로컬 환경 내부 디렉터리에 mount하고자 하였지만, 원하는대로 동작하지 않았습니다.
        - 따라서, Dockerfile 내부에 COPY 작업을 통해 npm run build로 생성된 정적 파일 디렉터리 dist를 복사 및 제공하였습니다.
    - /server/Dockerfile

```
FROM openjdk:17-slim-buster

RUN apt-get update && apt-get install -y ffmpeg

ARG JAR_FILE=build/libs/*.jar

ENV MONGO_USERNAME=leadme
ENV MONGO_PASSWORD=leadmessafy11

# 필요한 디렉토리 생성 (데이터 저장용)
RUN mkdir -p /home/ubuntu/python/video/temporary /home/ubuntu/python/vi

# 애플리케이션 코드 복사
COPY . .

# 지속적인 데이터 저장을 위해 볼륨 설정
VOLUME ["/home/ubuntu/python/video/temporary", "/home/ubuntu/python/vid

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java", "-jar", "/app.jar"]
```

        - 영상 데이터를 지속적으로 활용하고 저장하기 위해 디렉터리를 생성하고, ec2 로컬 환경 디렉터리에 볼륨을 설정하였습니다.

[외부 서비스 정보]

- 소셜 인증 관련 프로퍼티

```
spring.jwt.secret=vmfhaltmskdlstkfkdgodyroqkfwkdbalroqkfwkdbalaaaaaaaaaaaaa

#registration - naver
spring.security.oauth2.client.registration.naver.client-name=naver
spring.security.oauth2.client.registration.naver.client-id=ZYtCDpqybjyutk3
spring.security.oauth2.client.registration.naver.client-secret=qG1L4ZrPlD
spring.security.oauth2.client.registration.naver.redirect-uri=https://i11c
spring.security.oauth2.client.registration.naver.authorization-grant-type=
spring.security.oauth2.client.registration.naver.scope=name,email

#provider
spring.security.oauth2.client.provider.naver.authorization-uri=https://nid
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.c
```

```
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi
spring.security.oauth2.client.provider.naver.user-name-attribute=response

#registration - google
spring.security.oauth2.client.registration.google.client-name=google
spring.security.oauth2.client.registration.google.client-id=896191178237-l
spring.security.oauth2.client.registration.google.client-secret=GOCSPX-Apz
spring.security.oauth2.client.registration.google.redirect-uri=https://i11
spring.security.oauth2.client.registration.google.authorization-grant-type
spring.security.oauth2.client.registration.google.scope=profile,email

# spring oauth2 ???? ???? google provider? ??

#registration - kakao
spring.security.oauth2.client.registration.kakao.client-id=2a75df25c4dfed9
spring.security.oauth2.client.registration.kakao.client-secret=4rMlEM1RP4r
spring.security.oauth2.client.registration.kakao.redirect-uri=https://i11c
spring.security.oauth2.client.registration.kakao.authorization-grant-type=
spring.security.oauth2.client.registration.kakao.client-authentication-met
spring.security.oauth2.client.registration.kakao.scope=profile_nickname, p

# Kakao OAuth2 Provider
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kau
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.ka
spring.security.oauth2.client.provider.kakao.user-name-attribute=id

# id? ???? ?? crypto password
crypto.password=${crypto.password}
```