3. BE/FE Jenkins 스크립트 및 Docker 설정

9. BackEnd 배포 (dev-fe)

아래 Jenkins plugin 설치하기

Generic Webhook Trigger Plugin

GitLab API Plugin

GitLab Plugin

Stage View → 배포 과정 쉽게 볼 수 있음

9.1 app/docker-compose.yml 작성

```
services:
  app:
    build:
      context: ../../../backend/monthlyzip
      dockerfile: Dockerfile
      args:
        PROFILE: prod
    image: monthlyzip-app
    container_name: monthlyzip-app
    ports:
      - "8081:8081"
    environment:
      - SPRING_PROFILES_ACTIVE=prod
    volumes:
      - /home/ubuntu/images:/images  # 추가 : 이미지 디렉토리 마운트
    restart: always
    networks:
      - app-network
networks:
  app-network:
    external: true
```

9.2. DockerFile 작성

```
# backend/monthlyzip/Dockerfile

FROM amazoncorretto:17
ARG JAR_FILE=./build/libs/monthlyzip-0.0.1-SNAPSHOT.jar
ARG PROFILE
ENV SPRING_PROFILES_ACTIVE=${PROFILE}
WORKDIR /app
COPY ${JAR_FILE} app.jar
COPY src/main/resources/application-secret.yml /app/application-secret.yml
ENTRYPOINT ["java", "-Dspring.profiles.active=${SPRING_PROFILES_ACTIVE
```

9.3. Jenkins pipeline을 이용한 자동배포 스크립트

```
pipeline {
    agent any
    stages {
        stage('Check Branch') {
            steps {
                script {
                    // def targetBranch = 'dev-be'
                    def targetBranch = env.gitlabTargetBranch ?: env.BRANCH_NAME
                    if (targetBranch != null && targetBranch != 'dev-be') {
                        currentBuild.result = 'ABORTED'
                        error("This pipeline only runs for merge requests to dev-be bra
                    }
                }
            }
        }
        stage('CheckOut') {
            steps {
                echo 'Start CheckOut monthlyzip project...'
                git branch: 'dev-be',
                    credentialsId: 'account',
                    url: 'https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21D109.g
                            sh 'ls -R infra/docker/app'  // 파일 목록 출력
                sh '''
                pwd
                ls -R
              '''
          echo 'CheckOut finished!'
        }
      }
      stage('Build') {
        steps {
          echo 'Start building monthlyzip project...'
          dir('backend/monthlyzip') {
            withCredentials([file(credentialsId: 'application-secret', variable: 'S
              sh """
                cp -f "$SECRET_FILE" src/main/resources/application-secret
                cat src/main/resources/application-secret.yml
              """
            }
            sh '''
              chmod +x ./gradlew
              ./gradlew clean build -x test
            '''
          }
          echo 'Build finished!'
        }
      }

      stage('Deploy') {
        steps {
          script {
            dir('infra/docker/app') {
              // application-secret.yml 파일 복사
              withCredentials([file(credentialsId: 'application-secret', variable
                sh """
                  cp -f "\$SECRET_FILE" application-secret.yml
                  chmod 600 application-secret.yml
                """
              }

              // 기존 컨테이너 중지 및 제거
```

```
                    sh "docker compose down app || true"

                    // 새 이미지 빌드 및 컨테이너 시작
                    sh "docker compose build --no-cache app"
                    sh "docker compose up -d app"

                    sh "sleep 20" // 서버가 완전히 시작될 때까지 잠시 대기

                    // 배포 완료 후 시크릿 파일 삭제
                    sh "rm -f application-secret.yml"

                    echo 'Deploy finished!'
                }
            }
        }
    }

  }
  post {
    success {
      echo 'Pipeline succeeded!'
    }
    failure {
      echo 'Pipeline failed!'
      dir('infra/docker/app') {
        sh 'docker compose logs app'
      }
    }
    always {
      echo 'Cleaning up workspace'
      cleanWs()
    }
  }
}
```

10. FrontEnd 배포 (dev-fe)

아래 Jenkins plugin 설치하기

NodeJs Plugin

10.1. frontend/docker-compose.yml 작성

```
services:
  react:
    image: react
    build:
      context: ../../../frontend/monthly-zip
      dockerfile: Dockerfile
    container_name: react
    ports:
      - "3000:80"
    networks:
      - app-network
    restart: always

networks:
  app-network:
    external: true
```

10.2. DockerFile 작성

```
# 빌드 단계
FROM node:22.13.0-alpine as builder

# 작업 디렉토리 설정
WORKDIR /app

# 의존성 설치
COPY package*.json ./
RUN npm install

# 앱 소스 복사 및 빌드
COPY . .
RUN npm run build
```

```
# 프로덕션 단계
FROM nginx:alpine

# 빌드 결과물 복사
COPY --from=builder /app/build /usr/share/nginx/html

# 포트 설정
EXPOSE 80

# 실행 명령어
CMD ["nginx", "-g", "daemon off;"]
```

10.3. Nginx.conf 작성

```
# frontend/mafia/nginx.conf
server {
    listen 80;
    location / {
        root /usr/share/nginx/html;
        try_files $uri $uri/ /index.html;
    }
}
```

10.4. Jenkins pipeline를 이용하여배포

```
pipeline {
    agent any

    stages {
        stage('FE-Check Branch') {
            steps {
                script {
                    // def targetBranch = 'dev-fe'
                    def targetBranch = env.gitlabTargetBranch ?: env.BRANCH_NAME
                    if (targetBranch != null && targetBranch != 'dev-fe') {
                        currentBuild.result = 'ABORTED'
                        error("This pipeline only runs for merge requests to dev-fe bran
                    }
                }
            }
        }

        stage('FE-CheckOut') {
            steps {
                echo 'Start CheckOut monthlyzip project...'
                git branch: 'dev-fe',
                    credentialsId: 'account',
                    url: 'https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21D109.g
                echo 'CheckOut finished!'
            }
        }

        stage('FE-Build') {
            steps {
                echo 'Start building monthlyzip project...'
                nodejs(nodeJSInstallationName: 'NodeJS 22.13.0') {
                    dir('frontend/monthly-zip') {
                        sh '''
                            npm install
                            CI=false npm run build
                        '''

                    }
                }
                echo 'Build finished!'
            }
        }

        stage('FE-Deploy') {
            steps {
                script {
                    dir('infra/docker/frontend') {
                        sh '''
                            docker compose down || true
                            docker compose build --no-cache
```

```
                        docker compose up -d
                    '''

                    sh """
                        echo "Reloading Nginx configuration..."
                        docker exec nginx nginx -s reload
                        echo "Nginx reloaded successfully"
                    """

                    // 컨테이너 상태 확인
                    sh '''
                        echo "Waiting for containers to start..."
                        sleep 30
                        docker ps
                    '''
                }
            }
        }
    }
}

post {
    success {
        echo 'Frontend pipeline successful !!'
    }
    failure {
        echo 'Frontend pipeline failed !!'
        dir('infra/docker/frontend') {
            sh 'docker compose logs'
        }
    }
}
}
```