Auto Store 외부 서비스 정리

Jenkins Pipeline

Frontend

```
pipeline {
    agent any

    environment {
        DOCKER_HUB_REPO = 'junhyeok302/wms-frontend' // Docker Hub에 저장되는 위
        DOCKER_CREDENTIALS_ID = 'DockerHub'  // Docker Hub 인증 정보 ID
    }

    stages {
        stage('Clone') {
            steps {
                script {
                    git branch: 'dev/frontend/master', url: '<https://lab.ssafy.com/s11-ai-image-
sub1/S11P21A302.git>', credentialsId: 'Gitlab'
                }
            }
        }


        stage('Build') {
            steps {
                    dir('wms-front'){
                        sh 'npm install'
                        sh 'npm run build'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    dir('wms-front') {
                        def imageTag = "${DOCKER_HUB_REPO}:${env.BUILD_ID}"
                        sh "docker build -t ${imageTag} -f Dockerfile ."
                        env.DOCKER_IMAGE = imageTag
                    }
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    def imageTag = "${DOCKER_IMAGE}"
                    withCredentials([usernamePassword(credentialsId: DOCKER_CREDENTIALS_ID,
passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                        sh """
                        echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin
                        docker push ${imageTag}
                        docker tag ${imageTag} ${DOCKER_HUB_REPO}:latest
                        docker push ${DOCKER_HUB_REPO}:latest
                        """
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    def isBlue = sh(script: "docker ps | grep blue-fe", returnStatus: true) == 0

                    def newPort = isBlue ? 3002 : 3001
                    def oldPort = isBlue ? 3001 : 3002
                    def newContainer = isBlue ? 'green-fe' : 'blue-fe'
                    def oldContainer = isBlue ? 'blue-fe' : 'green-fe'

                    // 새로운 컨테이너 시작
                    sh """
                        docker pull ${DOCKER_HUB_REPO}:latest
                        docker stop ${newContainer} || true
                        docker rm ${newContainer} || true
                        docker run -d --name ${newContainer} -p ${newPort}:3000 ${DOCKER_IMAGE}
                    """

                    // NGINX 설정 파일 업데이트 (프론트엔드에 맞춰 포트 업데이트)
                    lock('nginx-config') {  // NGINX 설정 파일에만 락 적용
                        sh """
                        sudo sed -i 's#server localhost:300[12]#server localhost:${newPort}#'
/etc/nginx/sites-available/default
                        sudo nginx -t && sudo systemctl reload nginx
                        """
                    }

                    // 이전 컨테이너 종료 및 제거
                    sh """
                        docker stop ${oldContainer} || true
                        docker rm ${oldContainer} || true
                    """
                }
            }
        }
    }

    post {
        success {
                script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}
(${Author_Name})\\n(<${env.BUILD_URL}|Details>)",
                endpoint: '<https://meeting.ssafy.com/hooks/rt6i8ema3prg9nbckij1e4d8re>',
                channel: 'a302_jenkins'
                )
            }
        }
        failure {
                script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}
(${Author_Name})\\n(<${env.BUILD_URL}|Details>)",
                endpoint: '<https://meeting.ssafy.com/hooks/rt6i8ema3prg9nbckij1e4d8re>',
                channel: 'a302_jenkins'
                )
            }
        }
    }
}
```

Backend

```
pipeline {
    agent any

    environment {
        DOCKER_HUB_REPO = 'junhyeok302/wms-backend'
        CURRENT_SERVER_FILE = '/var/lib/jenkins/current_server_file'
        DOCKER_CREDENTIALS_ID = 'DockerHub'  // Docker Hub 인증 정보 ID
    }

    stages {
        stage('Clone') {
            steps {
                script {
                    git branch: 'dev/backend/master', url: '<https://lab.ssafy.com/s11-ai-image-
sub1/S11P21A302.git>', credentialsId: 'Gitlab'
                }
            }
        }
        stage('Prepare Info Properties') {
            steps {
                withCredentials([
                    string(credentialsId: 'dbUsername', variable: 'DB_USERNAME'),
                    string(credentialsId: 'dbPassword', variable: 'DB_PASSWORD'),
                    // string(credentialsId: 'datasourceUrl', variable: 'DB_URL'),
                    string(credentialsId: 'mySqlDocker', variable: 'DB_URL'),
                    string(credentialsId: 'KakaoClientId', variable: 'KAKAO_CLIENT_ID'),
                    string(credentialsId: 'kakaoClientSecret', variable: 'KAKAO_CLIENT_SECRET'),
                    string(credentialsId: 'naverClientId', variable: 'NAVER_CLIENT_ID'),
                    string(credentialsId: 'naverClientSecret', variable: 'NAVER_CLIENT_SECRET'),
                    string(credentialsId: 'redisHost', variable: 'REDIS_HOST'),
                    string(credentialsId: 'redisPort', variable: 'REDIS_PORT'),
                    string(credentialsId: 'redisPassword', variable: 'REDIS_PASSWORD'),
                    string(credentialsId: 'redisUsername', variable: 'REDIS_USERNAME'),
                    string(credentialsId: 'awsAccessKeyId', variable: 'AWS_ACCESS_KEY_ID'),
                    string(credentialsId: 'awsSecretKey', variable: 'AWS_SECRET_KEY'),
                ]) {
                    script {
                        dir('wms_backend/src/main/resources') {  // info.properties 파일을 생성할 디렉토리
로 이동
                            writeFile file: 'info.properties', text: """
                                spring.datasource.url=${DB_URL}
                                spring.datasource.username=${DB_USERNAME}
                                spring.datasource.password=${DB_PASSWORD}
                                spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
                                spring.jpa.hibernate.ddl-auto=update
                                spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
                                spring.jpa.show-sql=true

                                spring.data.redis.host=${REDIS_HOST}
                                spring.data.redis.port=${REDIS_PORT}
                                spring.data.redis.password=${REDIS_PASSWORD}
                                spring.data.redis.username=${REDIS_USERNAME}

                                spring.security.oauth2.client.registration.kakao.client-
id=${KAKAO_CLIENT_ID}
                                spring.security.oauth2.client.registration.kakao.client-
secret=${KAKAO_CLIENT_SECRET}

                                spring.security.oauth2.client.registration.naver.client-
id=${NAVER_CLIENT_ID}
                                spring.security.oauth2.client.registration.naver.client-
secret=${NAVER_CLIENT_SECRET}

                                aws.accessKeyId=${AWS_ACCESS_KEY_ID}
                                aws.secretKey=${AWS_SECRET_KEY}
                                aws.s3.bucket=a302
                                aws.region=ap-northeast-2

                            """
                        }
                    }
                }
            }
        }
        stage('Build') {
            steps {
                script {
                    dir('wms_backend') {  // 프로젝트의 루트 디렉토리로 이동
                        sh 'chmod +x ./gradlew'  // gradlew 파일에 실행 권한 부여
                        sh './gradlew clean build'
                    }
                }
            }
        }
        stage('Verify Build Artifacts') {
            steps {
                script {
                    dir('wms_backend') {
                        sh 'ls -la build/libs'  // build/libs 디렉토리 확인
                    }
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    dir('wms_backend') {  // 프로젝트의 Dockerfile이 있는 디렉토리로 이동
                        def imageTag = "${DOCKER_HUB_REPO}:${env.BUILD_ID}"
                        sh "ls -la build/libs"  // Docker 빌드 전에 build/libs 디렉토리 확인
                        sh "docker build -t ${imageTag} -f Dockerfile ."
                    }
                }
            }
        }
        stage('Push Docker Image') {
            steps {
                script {
                    def imageTag = "${DOCKER_HUB_REPO}:${env.BUILD_ID}"
                    withCredentials([usernamePassword(credentialsId: DOCKER_CREDENTIALS_ID,
passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                        sh """
                        echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin
                        docker push ${imageTag}
                        docker tag ${imageTag} ${DOCKER_HUB_REPO}:latest
                        docker push ${DOCKER_HUB_REPO}:latest
                        """
                    }
                }
            }
        }

        stage('Check and Create Current Server File') {
            steps {
                script {
                    // 파일이 없으면 'blue' 서버로 초기화
                    if (!fileExists(CURRENT_SERVER_FILE)) {
                        sh "echo 'blue' > ${CURRENT_SERVER_FILE}"
                    }
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    def currentServer = sh(script: "cat ${CURRENT_SERVER_FILE}", returnStdout:
true).trim()
                    def nextServer = currentServer == 'blue' ? 'green' : 'blue'

                    // Define ports for blue and green instances
                    def nextPort = nextServer == 'blue' ? '8081' : '8082'

                    // Start the new instance on a different port
                    sh """
                    docker pull ${DOCKER_HUB_REPO}:latest
                    docker stop ${nextServer}-be || true
                    docker rm ${nextServer}-be || true
                    docker run -d --name ${nextServer}-be --network backend-network -p ${nextPort}:8080
${DOCKER_HUB_REPO}:latest
                    echo ${nextServer} > ${CURRENT_SERVER_FILE}
                    """

                    // NGINX 설정 파일 업데이트에만 잠금 설정
                    lock('nginx-config') {  // NGINX 설정 파일에만 락을 걸어 동시 접근 방지
                        sh """
                        sudo sed -i 's#server localhost:808[12]#server localhost:${nextPort}#'
/etc/nginx/sites-available/default
                        sudo nginx -t && sudo systemctl reload nginx
                        """
                    }

                    // Stop and remove the current server
                    def currentPort = currentServer == 'blue' ? '8081' : '8082'
                    sh """
                    docker stop ${currentServer}-be || true
                    docker rm ${currentServer}-be || true
                    """
                }
            }
        }

    }

    post {
        success {
                script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}
(${Author_Name})\\n(<${env.BUILD_URL}|Details>)",
                endpoint: '<https://meeting.ssafy.com/hooks/rt6i8ema3prg9nbckij1e4d8re>',
                channel: 'a302_jenkins'
                )
            }
        }
        failure {
                script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}
(${Author_Name})\\n(<${env.BUILD_URL}|Details>)",
                endpoint: '<https://meeting.ssafy.com/hooks/rt6i8ema3prg9nbckij1e4d8re>',
                channel: 'a302_jenkins'
                )
            }
        }
    }
}
```