Memozy 포팅 메뉴얼

생성일	@2025년 5월 21일 오후 3:23
태그	포팅메뉴얼

1. 개요

본 문서는 Memozy 프로젝트의 배포 환경을 구성하기 위한 포팅 메뉴얼입니다.

Docker 기반의 컨테이너 환경에서 Jenkins, Nginx, Database, Spring Boot, React 등을 포함한 전체 시스템을 구성하며, CI/CD 파이프라인 설정까지 다루고 있습니다.

2. 배포환경 + 기술 스택

COMMON

- Jenkins
- Nginx
- Docker / Docker Compose
- Redis
- MySQL 8
- Ubuntu 22.04
- SonarQube

BACKEND

- JDK 17
- Gradle 8.14
- Spring Boot 3.4.3

FRONTEND

- React 18
- Node.js 22
- Vite

3. 환경 변수 설정 파일 목록

Frontend

- front/web/.env.production

```
VITE_API_BASE_URL=https://memozy.site
```

- front/web/.env.development

```
VITE_API_BASE_URL=https://test.memozy.site
```

Backend

- back/memozy-api/Dockerfile

```
FROM openjdk:17

WORKDIR /usr/src/app

ARG JAR_PATH=./build/libs
ARG JAR_NAME=memozy-api-0.0.1-SNAPSHOT.jar

COPY ${JAR_PATH}/${JAR_NAME} app.jar

ENV SPRING_PROFILES_ACTIVE=local

CMD ["java", "-jar", "-Dspring.profiles.active=${SPRING_PROFILES_ACTIVE}",
```

- back/memozy-api/src/main/resources/application-API-KEY.properties

```
# LOCAL----------------------------------------------
# DATABASE - LOCAL
LOCAL_DB_URL=jdbc:mysql://localhost:3306/test?serverTimezone=Asia/Seou
LOCAL_DB_USERNAME=root
LOCAL_DB_PASSWORD=root
# DATABASE - DEV
DEV_DB_URL=
DEV_DB_USERNAME=
```

```
DEV_DB_PASSWORD=
DEV_REDIS_PORT=
DEV_REDIS_HOST=
DEV_REDIS_PASSWORD=
# DATABASE - PROD
DEV_DB_URL=
DEV_DB_USERNAME=
PROD_DB_PASSWORD=
PROD_REDIS_PORT=
PROD_REDIS_HOST=
PROD_REDIS_PASSWORD=
# Google
## Google registration Web
spring.security.oauth2.client.registration.google.client-name=Google
spring.security.oauth2.client.registration.google.client-id=
spring.security.oauth2.client.registration.google.client-secret=
spring.security.oauth2.client.registration.google.authorization-grant-type=aut
spring.security.oauth2.client.registration.google.scope=profile,email
## Google provider
spring.security.oauth2.client.provider.google.authorization-uri=https://accoun
spring.security.oauth2.client.provider.google.token-uri=https://oauth2.google
spring.security.oauth2.client.provider.google.user-info-uri=https://openidconn
spring.security.oauth2.client.provider.google.user-name-attribute=sub
# JWT SECRET KEY
spring.jwt.secret=
# Open AI
spring.ai.openai.api-key=
spring.ai.openai.chat.options.model=gpt-4o
# Local
LOCAL_REDIRECT_URL=
# DEV
DEV_REDIRECT_URL=
# PROD
PROD_REDIRECT_URL=
# CHROME EXTENSION
LOCAL_CHROME_EXTENSION_URL=
DEV_CHROME_EXTENSION_URL=
PROD_CHROME_EXTENSION_URL=
```

4. 서버 설정

포트 설정

```
sudo ufw allow 22, 44, 80, 443, 3306, 8080, 9000, 9090, 9091
```

5. 필요한 리소스 설치

`mkdir memozy && cd memozy`

5.1 Docker & Docker Compose

```
# 1. 패키지 업데이트
sudo apt-get update

# 2. https 관련 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-c

# 3. 기존 Docker GPG 키 및 저장소 삭제
sudo rm -rf /etc/apt/keyrings/docker.gpg
sudo rm -f /etc/apt/sources.list.d/docker.list

# 4. 최신 Docker GPG 키 추가
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearm
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# 5. Docker 저장소 추가
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://dov
```

```
# 6. 패키지 목록 업데이트
sudo apt update

# 7. Docker 설치 (선택 사항)
sudo apt-get install docker-ce docker-ce-cli containerd.io -y

# 8. 설치 확인
docker --version
```

5.2 docker-compose.infra.yml

`mkdir -p memozy/infra && cd memozy/infra`

- vim docker-compose.infra.yml

```
services:

  jenkins:
    image: jenkins/jenkins:lts-jdk17
    container_name: jenkins
    networks:
      - app-network
    environment:
      - JENKINS_OPTS=--prefix=/jenkins
      - JAVA_OPTS=-Djenkins.model.Jenkins.crumbIssuerProxyCompatibility=tr
      - JENKINS_URL=https://memozy.site/jenkins
    volumes:
      - jenkins_data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    restart: unless-stopped
    ports:
      - "9090:8080"
      - "50000:50000"
```

```
    user: root

  nginx:
    image: nginx:alpine
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/logs:/var/log/nginx
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
      - /var/www/letsencrypt:/var/www/letsencrypt
      - /etc/letsencrypt:/etc/letsencrypt
      - /home/ubuntu/memozy/frontend/build:/usr/share/nginx/html
    networks:
      - app-network
    restart: unless-stopped



networks:
  app-network:
    name: app-network
    driver: bridge
    external: true

volumes:
  jenkins_data:
```

- vim nginx.conf

```
access_log  /var/log/nginx/access.log  combined;
error_log   /var/log/nginx/error.log   warn;
```

```
upstream spring_backend {
    server memozy-app-prod:8080;
    # server 43.203.212.111:8080;
}


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

        server_name memozy.site www.memozy.site k12a602.p.ssafy.io;

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

        location / {
                return 301 https://$host$request_uri;
        }
}

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

        resolver 127.0.0.11 ipv6=off;

        root /usr/share/nginx/html;
        index index.html;

        server_name memozy.site www.memozy.site k12a602.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/memozy.site/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/memozy.site/privkey.pem;


        location / {
```

```
        try_files $uri /index.html;
    }

    # jenkins
    location /jenkins {

        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_for;
        proxy_set_header    X-Forwarded-Proto $scheme;
        proxy_set_header    X-Forwarded-Host $host;
        proxy_set_header    X-Forwarded-Port 443;
        proxy_pass_header   Authorization;
        proxy_set_header    Authorization $http_authorization;
        proxy_set_header    X-NginX-Proxy true;

        proxy_pass          http://jenkins:8080/jenkins/;
        proxy_redirect      ~^/jenkins/jenkins /jenkins;
        proxy_cookie_path   ~^/jenkins/jenkins /jenkins;
        proxy_buffering     off;
        proxy_request_buffering off;
    }

    # 웹소켓
    location /ws-connect {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Authorization $http_authorization;
        proxy_set_header Host $host;

        access_log /var/log/nginx/ws_connect.log;

        proxy_read_timeout 3600;
```

```
        proxy_send_timeout 3600;

        proxy_pass http://spring_backend/ws-connect;
    }



    # 일반 백엔드 엔드포인트
    location ~ ^/(oauth2|login|api)/ {
            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;
            proxy_set_header    Authorization $http_authorization;
            proxy_set_header    X-NginX-Proxy true;

            proxy_pass          http://spring_backend$request_uri;
            proxy_redirect      off;
            charset             utf-8;
    }

    location /swagger-ui/ {
        proxy_pass http://spring_backend/swagger-ui/;
        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;
    }

    location /v3/api-docs {
        proxy_pass http://spring_backend/v3/api-docs;
        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;
    }

    location /swagger-config {
```

```
            proxy_pass http://spring_backend/swagger-config;
            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;
        }
}
```

`docker compose -f docker-compose.infra.yml up -d`

5.3 docker-compose.database.yml

`mkdir -p memozy/database && cd memozy/database`

- vim docker-compose.db.yml

```
services:

  memozy_prod:
    image: mysql:8.0
    container_name: memozy_prod
    restart: always
    env_file:
      - ./memozy_prod.env
    ports:
      - "3306:3306"
    command: --server-id=1 --log-bin=mysql-bin
    volumes:
      - memozy_prod_data:/var/lib/mysql
    networks:
      - app-network

  redis_prod:
    image: redis:latest
    container_name: redis_prod
    restart: always
```

```
    networks:
      - app-network
    ports:
      - "6380:6379"
    volumes:
      - redis_prod_data:/data
    command: [
      "redis-server",
      "--requirepass", "",
      "--bind", "0.0.0.0",
      "--protected-mode", "no"
    ]


networks:
  app-network:
    external: true
    name: app-network

volumes:
  memozy_prod_data:
  redis_prod_data:
```

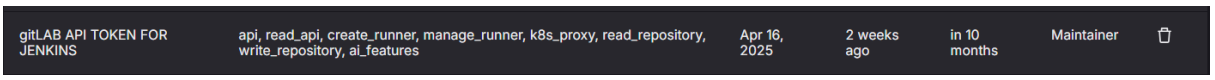
- vim memozy_prod.env

```
MYSQL_ROOT_PASSWORD: MYSQL_DATABASE:
MYSQL_USER:
MYSQL_PASSWORD:
```

`docker compose -f docker-compose.db.yml up -d`

6. Jenkins System, Plugin, Tools, Credentials

6.1 GitLab AccessToken



6.2 Jenkins Plugin

SSH Agent Plugin: Git 저장소, 원격 서버를 SSH 키로 연결

Pipreline: Declarative: pipeline 문법을 사용할 수 있도록 해줌

Pipeline: Stage View Plugin: Stage 상태를 시각적으로 보여줌

SonarQube Scanner: SonarQube를 사용할 수 있는 플러그인

Pipeline Utility Steps: SonarQube에서 받은 readJSON을 사용하기 위한 플러그인

Gradle Plugin: Gradle 빌드를 쉽게 실행할 수 있음

Docker plugin: CI/CD 파이프라인에서 컨테이너 기반 빌드 및 배포

NodeJs Plugin: npm/yam 명령어를 실행할 수 있음

MatterMostNotifier: Mattermost 웹훅을 통해 알림으로 전송

Workspace Cleanup: cleanWs 메서드를 사용하여 워크스페이스를 지우는 용도

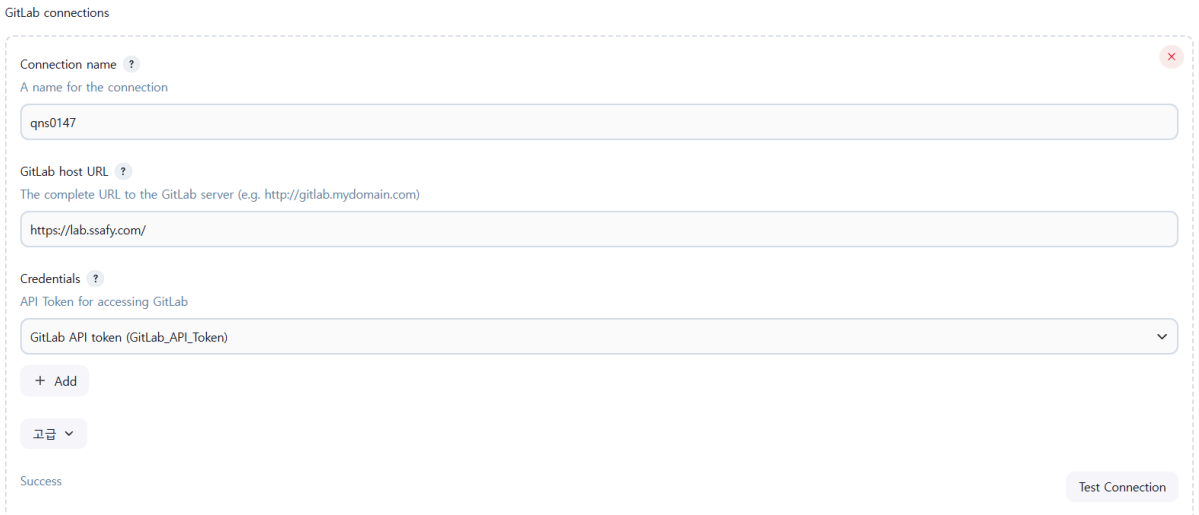
GitLab Plugin: GitLab의 이벤트(웹훅)를 감지

GitLab API: GitLab의 정보를 가져오거나 업데이트할 때 사용

6.3 Jenkins Credentials

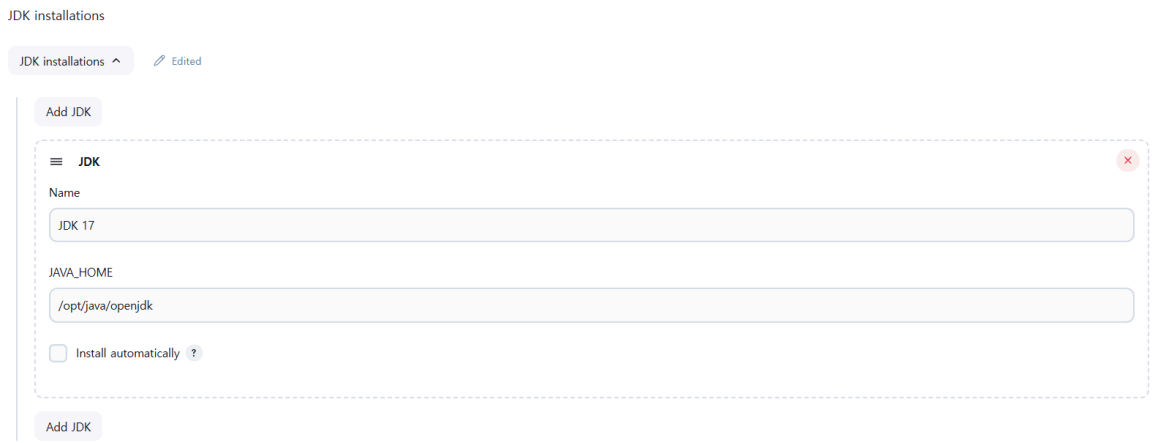
T	P	Store ↓	Domain	ID	Name
		System	(global)	Lee_GitLab_IdPassword	qns0147@gmail.com/****** (나의 깃랩 ID PS)
		System	(global)	GitLab_API_Token	GitLab API token (GitLab_API_Token)
		System	(global)	EC2_SSH	ubuntu (EC2_SSH)
		System	(global)	API_KEY_FILE	application-API-KEY.properties (API_KEY_FILE)
		System	(global)	docker-hub-credentials	qns0147@gmail.com/****** (docker-hub-credentials)
		System	(global)	SonarQubeCommenter	GitLab_Access_Token_For_SonarQube/****** (SonarQubeCommenter)
		System	(global)	sonarQubeToken	sonarQubeToken
		System	(global)	sonarprojectKey	sonarprojectKey
		System	(global)	sonarQubeFrontToken	sonarQubeFrontToken
		System	(global)	EC2_SSH_DEV	ubuntu (EC2_SSH_DEV)
		System	(global)	front-env	env.production (front-env)

6.4 Jenkins System

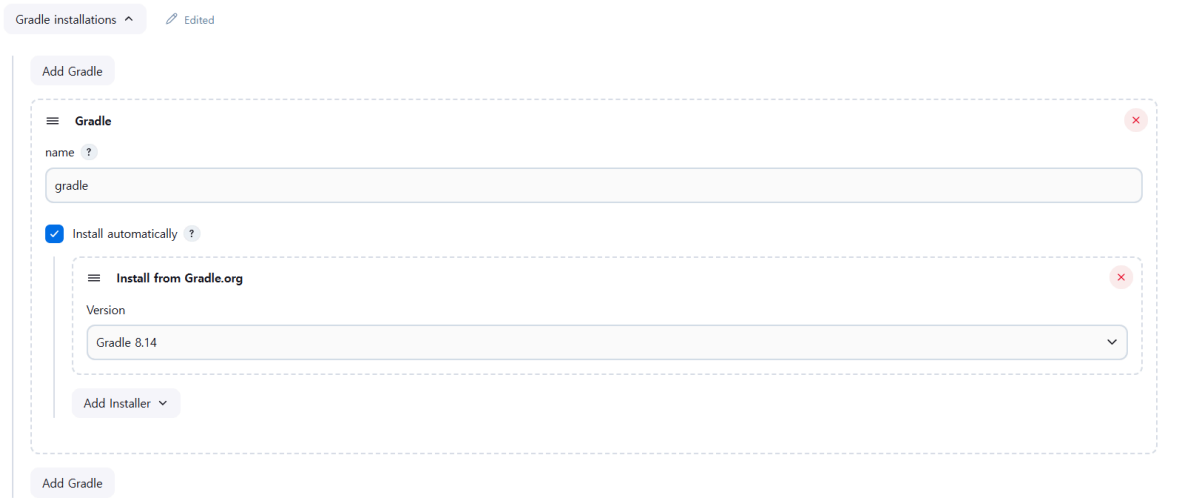


6.5 Jenkins Tools

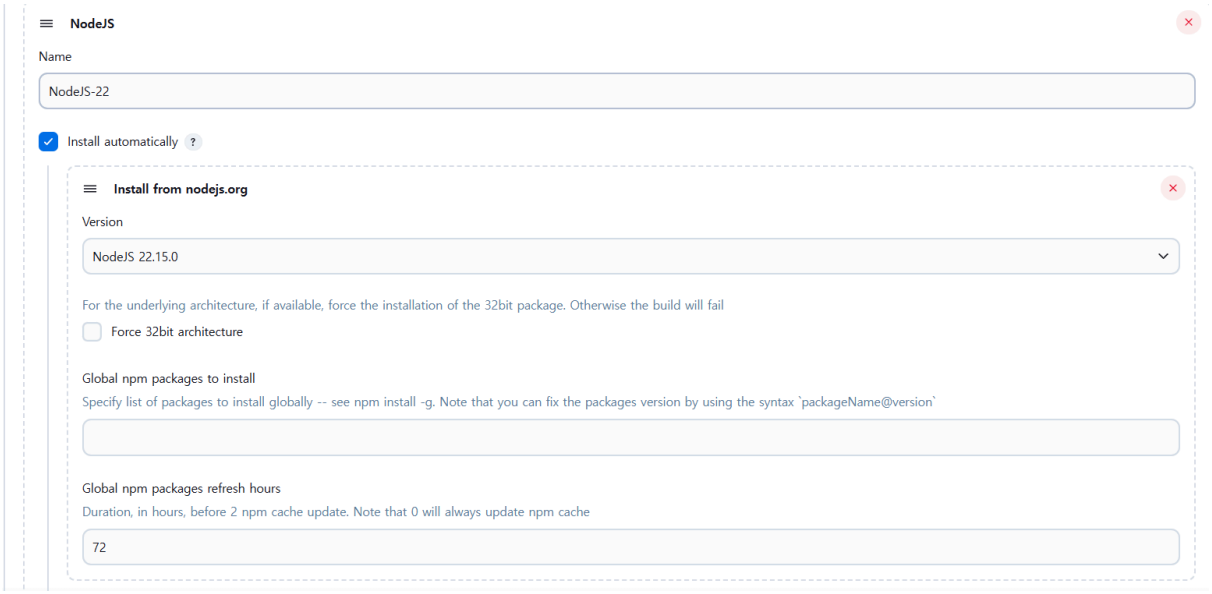
- JDK



- Gradle



- NodeJs



7. CI/CD Jenkins 파이프라인 스크립트

7.1 Front CI

```
pipeline {
    agent any
    environment {
        MATTERMOST_WEBHOOK_URL = ""
        sourceBranch = "${env.gitlabSourceBranch ?: 'front'}"
        targetBranch = "${env.gitlabTargetBranch ?: 'front'}"
        gitlabProjectId = "${env.gitlabProjectId}"
        gitlabMergeRequestIid = "${env.gitlabMergeRequestIid}"
    }
    tools {
        nodejs 'NodeJS-22'
    }

    stages {

        stage('Checkout Source') {
            steps {
```

```
            script {
                try {
                    updateGitlabCommitStatus name: 'checkout', state: 'pending'
                    echo "Checking out source branch: ${sourceBranch}"
                    git branch: "${sourceBranch}", credentialsId: '"
                    updateGitlabCommitStatus name: 'checkout', state: 'success'
                } catch (Exception e) {
                    updateGitlabCommitStatus name: 'checkout', state: 'failed'
                    throw e
                }
            }
        }
    }

    stage('Checkout Target') {
        steps {
            script {
                try {
                    echo "Checking out target branch: ${targetBranch}"
                    git branch: "${targetBranch}", credentialsId: 'Lee_GitLab_IdPass
                    updateGitlabCommitStatus name: 'checkout', state: 'success'
                } catch (Exception e) {
                    updateGitlabCommitStatus name: 'checkout', state: 'failed'
                    throw e
                }
            }
        }
    }

    stage('Merge Source into Target') {
        steps {
            script {
                try {
                    updateGitlabCommitStatus name: 'merge', state: 'pending'
                    sh '''
                    git config --global user.email "qns0147@gmail.com"
                    git config --global user.name "TinyFrogs"
                    '''
```

```
                echo "Merging ${sourceBranch} into ${targetBranch}"
                sh "git checkout ${targetBranch}"
                sh "git merge ${sourceBranch} --no-ff -m 'Merge ${sourceBran
                updateGitlabCommitStatus name: 'merge', state: 'success'

                updateGitlabCommitStatus name: 'check-conflicts', state: 'pend
                def mergeStatus = sh(script: 'git status', returnStdout: true).trim
                echo "Git status: ${mergeStatus}"

                if (mergeStatus.contains("Unmerged paths")) {
                    updateGitlabCommitStatus name: 'check-conflicts', state: 'fai
                    error "Merge conflict detected! Please resolve conflicts befor
                } else {
                    echo "Merge successful without conflicts."
                    updateGitlabCommitStatus name: 'check-conflicts', state: 'su
                }
            } catch (Exception e) {
                updateGitlabCommitStatus name: 'merge', state: 'failed'
                updateGitlabCommitStatus name: 'check-conflicts', state: 'failed
                throw e
            }
        }
    }
}

stage('Install Dependencies') {
    steps {
        script {
            try {
                updateGitlabCommitStatus name: 'Install', state: 'pending'
                sh '''
                    cd front/web
                    npm ci --legacy-peer-deps
                '''
                updateGitlabCommitStatus name: 'Install', state: 'success'
            } catch (Exception e) {
                updateGitlabCommitStatus name: 'Install', state: 'failed'
                throw e
```

```
            }
        }
    }
}

stage('Build & Test') {
    steps {
        script {
            try {
                updateGitlabCommitStatus name: 'build', state: 'pending'
                echo "Merge Request IID: ${gitlabMergeRequestIid}"
                sh '''
                    cd front/web
                    npm run build
                '''
                updateGitlabCommitStatus name: 'build', state: 'success'
            } catch (Exception e) {
                updateGitlabCommitStatus name: 'build', state: 'failed'
                throw e
            }
        }
    }
}
}

post {
    success {
        script {
            echo "✅ FRONT CI 성공! Mattermost로 알림 전송."
            sh """
                response=\$(curl -s -o response.txt -w "%{http_code}" -H "Content
                -X POST \\
                -d '{
                    "text": "✅ FRONT CI 성공!\\n🔗 Jenkins URL: ${env.BUILD_URL
                }' \\
                "$MATTERMOST_WEBHOOK_URL")

                echo "Mattermost Webhook Response Code: \$response"
```

```
            cat response.txt
        """
      }
    }
    failure {
      script {
        echo "❌ FRONT CI 실패! Mattermost로 알림 전송."
        sh """
          response=\$(curl -s -o response.txt -w "%{http_code}" -H "Content
            -X POST \\
            -d '{
              "text": "❌ FRONT CI 실패!\\n🔗 Jenkins URL: ${env.BUILD_URL
            }' \\
            "$MATTERMOST_WEBHOOK_URL")

          echo "Mattermost Webhook Response Code: \$response"
          cat response.txt
        """
      }
    }
}



}
```

7.2 Front CD

```
pipeline {
    agent any
    environment {
        MATTERMOST_WEBHOOK_URL = ""
        ENV = credentials('front-env')
    }
    tools {
        nodejs 'NodeJS-22'
    }
```

```
stages {
    stage('Cleanup Workspace') {
        steps {
            cleanWs()
        }
    }

    stage('Checkout') {
        steps {
            git branch: '', credentialsId: '', url: ""
        }
    }

    stage('Check for Changes in front/') {
        steps {
            script {
                def changes = sh(script: "git diff --name-only HEAD~1 | grep '^fro
                if (!changes) {
                    echo "🚀 No changes in front/ directory. Skipping pipeline."
                    currentBuild.result = 'ABORTED'
                    error("No changes detected in front/. Pipeline stopped.")
                } else {
                    echo "✅ Changes detected in front/: ${changes}"
                }
            }
        }
    }

stage('Copy ENV File') {
        steps {
            script {
                sh '''
                    cp "$ENV" front/web/.env.production
                '''
            }

        }
```

```
    }


    stage('Install Dependencies') {
        steps {
            script {
                sh '''
                    cd front/web
                    npm ci --legacy-peer-deps
                '''
            }
        }
    }

    stage('Build & Test') {
        steps {
            script {
                sh '''
                    cd front/web
                    npm run build
                '''
            }
        }
    }

    stage('Connect to EC2 and Deploy') {
        steps {
            script {
                sshagent(['EC2_SSH']) {  // Jenkins에 저장한 SSH Key ID
                    sh """
                    echo "✅ EC2에 파일 전송 시작!"
                    scp -o StrictHostKeyChecking=no -r
                    """
                }
            }
        }
    }
}
```

```
post {
    success {
        script {
            echo "✅ FRONT CD 성공! Mattermost로 알림 전송."
            sh """
                response=\$(curl -s -o response.txt -w "%{http_code}" -H "Content
                    -X POST \\
                    -d '{
                        "text": "✅ FRONT CD 성공!\\n🔗 Jenkins URL: ${env.BUILD_URL
                    }' \\
                    "$MATTERMOST_WEBHOOK_URL")

                echo "Mattermost Webhook Response Code: \$response"
                cat response.txt
            """
        }
    }
    failure {
        script {
            echo "❌ FRONT CD 실패! Mattermost로 알림 전송."
            sh """
                response=\$(curl -s -o response.txt -w "%{http_code}" -H "Content
                    -X POST \\
                    -d '{
                        "text": "❌ FRONT CD 실패!\\n🔗 Jenkins URL: ${env.BUILD_URL
                    }' \\
                    "$MATTERMOST_WEBHOOK_URL")

                echo "Mattermost Webhook Response Code: \$response"
                cat response.txt
            """
        }
    }
}

}
```

7.3 Back CI

```
pipeline {
    agent any

    environment {
        MATTERMOST_WEBHOOK_URL = ""
        API_KEY_FILE = credentials('API_KEY_FILE')
        SONAR_QUBE_TOKEN = credentials('sonarQubeToken')
        SONAR_PROJECT_KEY = credentials('sonarprojectKey')
        SONAR_HOST_URL = ""
        sourceBranch = "${env.gitlabSourceBranch ?: 'back'}"
        targetBranch = "${env.gitlabTargetBranch ?: 'back'}"
        gitlabProjectId = ""
        gitlabMergeRequestIid = "${env.gitlabMergeRequestIid}"
    }

    stages {
        stage('Checkout Source') {
            steps {
                script {
                    updateGitlabCommitStatus name: 'checkout', state: 'pending'

                    echo "Checking out source branch: ${sourceBranch}"
                    git branch: "${sourceBranch}", credentialsId: '', url: ""
                }
            }
        }

        stage('Checkout Target') {
            steps {
                script {
                    echo "Checking out target branch: ${targetBranch}"
                    git branch: "${targetBranch}", credentialsId: '', url: ""
```

```
            updateGitlabCommitStatus name: 'checkout', state: 'success'
        }
    }
}

stage('Merge Source into Target') {
    steps {
        script {
            updateGitlabCommitStatus name: 'merge', state: 'pending'

            sh '''
            git config --global user.email "@gmail.com"
            git config --global user.name ""
            '''

            echo "Merging ${sourceBranch} into ${targetBranch}"
            sh "git checkout ${targetBranch}"
            sh "git merge ${sourceBranch} --no-ff -m 'Merge ${sourceBranch

            updateGitlabCommitStatus name: 'merge', state: 'success'
            updateGitlabCommitStatus name: 'check-conflicts', state: 'pending

            def mergeStatus = sh(script: 'git status', returnStdout: true).trim()
            echo "Git status: ${mergeStatus}"

            if (mergeStatus.contains("Unmerged paths")) {
                updateGitlabCommitStatus name: 'check-conflicts', state: 'failed
                error "Merge conflict detected! Please resolve conflicts before p
            } else {
                echo "Merge successful without conflicts."
                updateGitlabCommitStatus name: 'check-conflicts', state: 'succ
            }
        }
    }
}

stage('Copy API Key File') {
```

```
        steps {
            script {
                updateGitlabCommitStatus name: 'copy api-key', state: 'pending'
                sh '''
                chmod -R u+w ./back/memozy-api/src/main/resources/
                cp $API_KEY_FILE ./back/memozy-api/src/main/resources/applica
                '''
                updateGitlabCommitStatus name: 'copy api-key', state: 'success'
            }
        }
    }

    stage('Build JAR') {
        steps {
            script {
                updateGitlabCommitStatus name: 'build', state: 'pending'
                sh '''
                cd back/memozy-api
                chmod +x gradlew
                ./gradlew build -x test
                '''
                updateGitlabCommitStatus name: 'build', state: 'success'
            }
        }
    }

  }

post {
  success {
    script {
      withEnv(["MATTERMOST_WEBHOOK_URL=${MATTERMOST_WEBHOC
        sh """
          curl -s -o response.txt -w "%{http_code}" -H "Content-Type: appl
            -X POST \\
            -d '{
              "text": "✅ BACK CI 성공!\\n🔗 Jenkins URL: ${env.BUILD_URL
            }' \\
```

```
                "\$MATTERMOST_WEBHOOK_URL"
            """
        }
    }
    failure {
        script {
            withEnv(["MATTERMOST_WEBHOOK_URL=${MATTERMOST_WEBHOC
                sh """
                    curl -s -o response.txt -w "%{http_code}" -H "Content-Type: appl
                        -X POST \\
                        -d '{
                            "text": "❌ BACK CI 실패!\\n🔗 Jenkins URL: ${env.BUILD_URL
                        }' \\
                        "\$MATTERMOST_WEBHOOK_URL"
                """
            }
        }
    }
}


}
```

7.4 Back CD

```
pipeline {
    agent any

    environment {
        MATTERMOST_WEBHOOK_URL = ""
        API_KEY_FILE = credentials('API_KEY_FILE')
        IMAGE_NAME = ""
        IMAGE_TAG = ""
        DOCKER_REPO = ""
```

```
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: '', credentialsId: '', url: ""
            }
        }

        stage('Copy API Key File') {
            steps {
                script {
                    sh '''
                    chmod -R u+w ./back/memozy-api/src/main/resources/
                    cp $API_KEY_FILE ./back/memozy-api/src/main/resources/applica
                    '''
                }
            }
        }

        stage('Build JAR') {
            steps {
                script {
                    echo "✅ Gradle 프로젝트 빌드 시작"
                    echo "Setting JAVA_HOME to JDK 17"
                    sh '''
                    pwd
                    ls
                    cd back/memozy-api && chmod +x gradlew && ./gradlew build -x
                    '''
                    echo "✅ 빌드 완료"
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
```

```
                echo "✅ Docker 이미지 빌드 시작"
                sh '''
                export PATH=$PATH:/usr/bin
                ls
                cd back/memozy-api   # Dockerfile이 있는 경로로 이동
                docker build -t ${IMAGE_NAME}:${IMAGE_TAG} .
                '''
                echo "✅ Docker 이미지 빌드 완료: ${IMAGE_NAME}:${IMAGE_TAG}
            }
        }
    }

    stage('Push Docker Image') {
        steps {
            script {
                withCredentials([
                    usernamePassword(credentialsId: 'docker-hub-credentials', use
                ]) {
                    echo "✅ Docker Hub 로그인"
                    sh '''
                    echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_US
                    '''

                    echo "✅ Docker 이미지 태그 생성"
                    sh '''
                    docker tag "$IMAGE_NAME:$IMAGE_TAG" "$DOCKER_REPO:$I
                    '''

                    echo "✅ Docker Hub로 이미지 푸시"
                    sh '''
                    docker push "$DOCKER_REPO:$IMAGE_TAG"
                    '''
                }
            }
        }
    }

stage('Connect to EC2 and Deploy') {
```

```
    steps {
        script {
            sshagent(['EC2_SSH']) {
                sh '''#!/bin/bash
ssh -o StrictHostKeyChecking=no ubuntu@k12a602.p.ssafy.io << 'ENDSSH'
echo "✅ EC2에 접속 완료!"

docker stop memozy-app-prod || true
docker rm -f memozy-app-prod || true

docker pull tinyfrog/memozy-prod:latest

echo "🚀 도커 컨테이너 실행..."
docker run -d --name memozy-app-prod \
  --network app-network \
  -p 8080:8080 \
  -e SPRING_PROFILES_ACTIVE=prod \
  -v /home/ubuntu/memozy/server/logs:/var/log/memozy \
  tinyfrog/memozy-prod:latest

echo "✅ 배포 완료!"
ENDSSH
'''
            }
        }
    }
}




    }


    post {
    success {
        script {
            withEnv(["MATTERMOST_WEBHOOK_URL=${MATTERMOST_WEBHOC
```

```
            sh """
                curl -s -o response.txt -w "%{http_code}" -H "Content-Type: appl
                    -X POST \\
                    -d '{
                        "text": " ✅ BACK CD 성공!\\n🔗 Jenkins URL: ${env.BUILD_U
                    }' \\
                    "\$MATTERMOST_WEBHOOK_URL"
            """
        }
    }
}
failure {
    script {
        withEnv(["MATTERMOST_WEBHOOK_URL=${MATTERMOST_WEBHOC
            sh """
                curl -s -o response.txt -w "%{http_code}" -H "Content-Type: appl
                    -X POST \\
                    -d '{
                        "text": " ### ❌ BACK CD 실패!\\n🔗 Jenkins URL: ${env.BUIL
                    }' \\
                    "\$MATTERMOST_WEBHOOK_URL"
            """
        }
    }
}
}
```