Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1) 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전 (IDE 버전 포함)

Back-End

Java	17
Spring Boot	3.3.3
Redis	7.4.0
MySql	8.0.22
Nginx	1.27.1
Docker	27.3.1
Jenkins	2.478

Android

Android	13
Kotlin	1.9.22

AI

Kaggle	1.0
Python	3.10
TensorFlow Lite	2.1.0

2) 빌드 시 사용되는 환경 변수 내용 상세 기재

Android

- Pipeline File

```
pipeline {
    agent any
    environment {
        KIWI_ENV = credentials('jenkins_kiwi_android.env')
        JAVA_HOME = '/opt/java/openjdk'
        ANDROID_HOME = '/opt/android-sdk'
        PATH = "$JAVA_HOME/bin:$ANDROID_HOME/platform-tools:$PATH"
    }
    stages {
        stage('Load Environment Variables') {
            steps {
                script {
                    def props = readProperties file: "${KIWI_ENV}"
                    // Mattermost 관련 환경변수 추가
                    env.MATTERMOST_CHANNEL_NAME = props.MATTERMOST_CHANNEL_NAME
                    env.MATTERMOST_WEBHOOK_URL = props.MATTERMOST_WEBHOOK_URL
                    // 기존 환경변수들
                    env.DIST_PATH = props.DIST_PATH
                    env.REMOVE_PREFIX = props.REMOVE_PREFIX
                    env.HOST_PATH = props.HOST_PATH
```

```
            }
        }
    }
    stage('Checkout') {
        steps {
            checkout scm
        }
    }
    stage('Prepare Config') {
        steps {
            withCredentials([file(credentialsId: 'kiwi-android-properties', variable
                sh '''
                    chmod -R 755 KIWE-Android
                    cp $CONFIG_FILE KIWE-Android/local.properties
                '''
            }
        }
    }
    stage('Build') {
        steps {
            sh '''
                cd KIWE-Android
                chmod +x gradlew
                ./gradlew clean assembleDebug
            '''
        }
    }
    stage('Transfer') {
        steps {
            sshPublisher(
                publishers: [
                    sshPublisherDesc(
                        configName: 'D205-Server',
                        transfers: [
                            sshTransfer(
                                sourceFiles: '**/kiosk-debug.apk',
                                remoteDirectory: 'dev/nginx/android/apk',
                                removePrefix: 'KIWE-Android/app/kiosk/build/outputs/a
                            )
                        ],
                        usePromotionTimestamp: false,
                        useWorkspaceInPromotion: false
                    )
                ]
            )
        }
    }
}
post {
    always {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tr
            def Build_Status = currentBuild.result ?: 'SUCCESS'
            def Status_Color = Build_Status == 'SUCCESS' ? 'good' : (Build_Status ==
            def Status_Text = Build_Status == 'SUCCESS' ? '빌드 성공' : (Build_Status


            def message = """
                #### 👨‍🦱Android ${Status_Text}
```

```
                    **APK 다운로드:** [apk-debug](http://k11d205.p.ssafy.io/android/apk/ki
                    **빌드 URL:** [Details](${env.BUILD_URL})
                """.stripIndent()
                mattermostSend(
                    color: Status_Color,
                    message: message,
                    endpoint: "${env.MATTERMOST_WEBHOOK_URL}",
                    channel: "${env.MATTERMOST_CHANNEL_NAME}",
                    icon: 'https://jenkins.io/images/logos/jenkins/jenkins.png'
                )
            }
        }
    }
}
```

- local.properties

```
sdk.dir=(본인의 SDK 경로)
BASE_URL = "k11d205.p.ssafy.io/api/"
FAST_URL = "k11d205.p.ssafy.io"
BASE_IMAGE_URL = "k11d205.p.ssafy.io/api/static/"
KIOSK_ID = 51
```

- jenkions_kiwi_android.env

```
NGINX_CONTAINER_NAME=nginx
DIST_PATH="KIWE-Android/app/kiosk/build/outputs/apk/debug/kiosk-debug.apk"
REMOVE_PREFIX="KIWE-Android/app/kiosk/build/outputs/apk/debug/"
HOST_PATH="/home/ubuntu/dev/nginx/android/apk"
MATTERMOST_CHANNEL_NAME=d205-mr-alram
MATTERMOST_WEBHOOK_URL=https://meeting.ssafy.com/hooks/8b16opypkpnh7yizpcyurkg7cw
```

Backend

- Pipeline File

```
pipeline {
    agent any
    environment {
        DOCKERHUB_CREDENTIALS = credentials('docker_hub_token')
        KIWI_ENV = credentials('jenkins_kiwi_backend.env') // secretfile 가져오기
    }
    stages {
        stage('Load Environment Variables') {
            steps {
                script {
                    def props = readProperties(file: env.KIWI_ENV)
                    env.DOCKER_REPO = props['DOCKER_REPO']
                    env.DOCKER_TAG = props['DOCKER_TAG']
                    env.DOCKER_IMAGE = props['DOCKER_IMAGE']
                    env.BLUE_PORT = props['BLUE_PORT']
                    env.GREEN_PORT = props['GREEN_PORT']
                    env.CONTAINER_NAME_BLUE = props['CONTAINER_NAME_BLUE']
                    env.CONTAINER_NAME_GREEN = props['CONTAINER_NAME_GREEN']
                    env.CONTAINER_URL = props['CONTAINER_URL']
                    env.NGINX_CONTAINER_NAME = props['NGINX_CONTAINER_NAME']
                    env.NGINX_CONFIG_PATH = props['NGINX_CONFIG_PATH']
```

```
                    env.DEPLOY_LOG_PATH = props['DEPLOY_LOG_PATH']
                    env.MATTERMOST_CHANNEL_NAME = props['MATTERMOST_CHANNEL_NAME']
                    env.MATTERMOST_WEBHOOK_URL = props['MATTERMOST_WEBHOOK_URL']
                    env.DOCKER_NETWORK = props['DOCKER_NETWORK']
                }
            }
        }
        stage('Notify Build Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
                    def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout
                    def Commit_Message = sh(script: "git log -1 --pretty=%B", returnStdou

                    // 이전 커밋을 확인하고 기본값을 설정
                    def previousCommit = env.GIT_PREVIOUS_SUCCESSFUL_COMMIT ?: 'HEAD'
                    def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)'

                    // 커밋 메시지 이스케이프 처리
                    def formattedCommits = allCommits.split('\n').collect { line ->
                        // 이스케이프 문자를 제대로 처리하고, 필요없는 이스케이프 문자는 제거
                        def escapedLine = line.replaceAll("([\\[\\]\\(\\)])", '\\\\$1') /
                        "• ${escapedLine}"
                    }.join('\n')  // 실제 줄바꿈을 사용

                    def message = """
                        #### 🌐BE 빌드 시작
                        **빌드 번호:** $env.JOB_NAME #$env.BUILD_NUMBER
                        **브랜치:** $env.GIT_BRANCH
                        **작성자:** $Author_ID ($Author_Name)
                        **빌드 URL:** [Details]($env.BUILD_URL)
                        **포함된 커밋:**
                        $formattedCommits
                    """.stripIndent()

                    mattermostSend(
                        color: '#439FE0',
                        message: message,
                        endpoint: "$MATTERMOST_WEBHOOK_URL",
                        channel: "$MATTERMOST_CHANNEL_NAME",
                        icon: 'https://jenkins.io/images/logos/jenkins/jenkins.png'
                    )
                }
            }
        }
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Prepare Config') {
            steps {
                withCredentials([
                    file(credentialsId: 'kiwi-application-properties', variable: 'CONFIG_
                    file(credentialsId: 'kiwi-application-dev-properties', variable: 'CON
                ]) {
                    sh '''
                        mkdir -p KIWI-Backend/src/main/resources
                        chmod -R 755 KIWI-Backend/src/main/resources
                        cp $CONFIG_FILE KIWI-Backend/src/main/resources/application.prope
```

```
                    cp $CONFIG_DEV_FILE KIWI-Backend/src/main/resources/application-d
                '''
            }
        }
    }
    stage('Build & Test') {
        steps {
            sh 'cd KIWI-Backend && chmod +x gradlew && ./gradlew clean build'
        }
    }
    stage('Build Docker Image') {
        steps {
            script {
                sh 'cd KIWI-Backend && docker build -t $DOCKER_IMAGE .'
            }
        }
    }
    stage('Login to Docker Hub') {
        steps {
            script {
                withCredentials([usernamePassword(credentialsId: 'docker_hub_token',
                    sh 'docker login -u $DOCKER_HUB_CREDENTIALS_USR -p $DOCKER_HUB_CR
                }
            }
        }
    }
    stage('Push Docker Image') {
        steps {
            script {
                sh 'docker push $DOCKER_IMAGE'
            }
        }
    }
    stage('Deploy and Update Nginx') {
        steps {
            script {
                sshPublisher(
                    publishers: [
                        sshPublisherDesc(
                            configName: 'D205-Server',
                            transfers: [
                                sshTransfer(
                                    execCommand: """
                                        set -x  # 디버깅 모드 활성화
                                        exec > >(tee $DEPLOY_LOG_PATH) 2>&1
                                        docker pull $DOCKER_IMAGE

                                        # 새로 배포할 컨테이너 지정
                                        if [[ \$(docker ps -aq --filter name=$CONTAIN
                                            # 블루 컨테이너 상태를 확인
                                            CONTAINER_STATUS=\$(docker inspect -f '{

                                            if [[ "\$CONTAINER_STATUS" == "exited" ]]
                                                # exited 상태일 경우, 블루 컨테이너를 새 컨
                                                NEW_PORT=$BLUE_PORT
                                                NEW_NAME=$CONTAINER_NAME_BLUE
                                                OLD_NAME=$CONTAINER_NAME_GREEN
                                                CURRENT_PORT=$GREEN_PORT
                                                docker rm \$NEW_NAME
                                            else
```

```
                        # 그린 컨테이너를 새 컨테이너로 지정
                        NEW_PORT=$GREEN_PORT
                        NEW_NAME=$CONTAINER_NAME_GREEN
                        OLD_NAME=$CONTAINER_NAME_BLUE
                        CURRENT_PORT=$BLUE_PORT
                        docker rm \$NEW_NAME
                    fi
                fi

                # 새 이미지로 컨테이너 생성
                docker run -d --name \$NEW_NAME --network $DO

                sleep 30
                if curl -sf -m 10 http://localhost:\$NEW_PORT
                    # Update Nginx config
                    sed -i "s/server \$NEW_NAME:8080 down;/se
                    sed -i "s/server \$OLD_NAME:8080;/server
                    # Restart Nginx container
                    docker restart $NGINX_CONTAINER_NAME
                                        # Stop old container (not remove)
                                        docker stop \$OLD_NAME
                                        echo "Switched to new version on port \$N
                                        exit 0
                                    else
                                        docker stop \$NEW_NAME
                                        echo "Deployment failed, keeping old vers
                                        exit 1
                                    fi
                                """
                            )
                        ]
                    )
                ]
            )
        }
    }
}
post {
    always {
        script {
            def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true
            def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: tr
            def Commit_Message = sh(script: "git log -1 --pretty=%B", returnStdout: t
            def Build_Status = currentBuild.result ?: 'SUCCESS'
            def Status_Color = Build_Status == 'SUCCESS' ? 'good' : (Build_Status ==
            def Status_Text = Build_Status == 'SUCCESS' ? '빌드 성공' : (Build_Status

            // 이전 커밋을 확인하고 기본값을 설정
            def previousCommit = env.GIT_PREVIOUS_SUCCESSFUL_COMMIT ?: 'HEAD'
            def allCommits = sh(script: "git log --pretty=format:'%h - %s (%an)' ${pr
            def formattedCommits = allCommits.split('\\n').collect { line ->
                def escapedLine = line.replaceAll("([\\[\\]\\(\\)])", '\\\\$1')
                "• ${escapedLine}"
            }.join('\\n')

            def message = """
                #### 🌐BE $Status_Text
                **빌드 번호** $env.JOB_NAME #$env.BUILD_NUMBER
                **작성자:** $Author_ID ($Author_Name)
```

```
                    **빌드 URL:** [Details]($env.BUILD_URL)
                    **포함된 커밋:**
                    $formattedCommits
                """.stripIndent()
                mattermostSend(
                    color: Status_Color,
                    message: message,
                    endpoint: "$env.MATTERMOST_WEBHOOK_URL",
                    channel: "$env.MATTERMOST_CHANNEL_NAME",
                    icon: 'https://jenkins.io/images/logos/jenkins/jenkins.png'
                )
            }
        }
    }
}
```

- application.properties

```
spring.application.name=KIWI-Backend
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.data.redis.repositories.enabled=true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://k11d205.p.ssafy.io:3306/kiwi
spring.datasource.username=kiwi
spring.datasource.password=wiki
spring.jpa.properties.hibernate.format_sql=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.cache.type=redis
spring.profiles.include=dev
spring.profiles.active=dev
spring.servlet.encoding.charset=UTF-8
spring.servlet.encoding.enabled=true
spring.servlet.encoding.force=true
jwt.secret-key=NiOeyFbN1Gqo10bPgUyTFsRMkJpGLXSvGP04eFqj5B30r5TcrtlSXfQ7TndvYjNvfkEKLqILnC
jwt.issuer=colabear754
jwt.expiration-minutes=30
refresh-expiration-hours=168

spring.mvc.static-path-pattern=/static/**
spring.web.resources.static-locations=classpath:/static/,file:/var/images/
server.image.base-url=http://localhost:8080/static/

server.forward-headers-strategy=FRAMEWORK



# Swagger UI ?? ??
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.swagger-ui.operations-sorter=method

# API Docs ?? ??
springdoc.api-docs.path=/v3/api-docs

# ?? ??? ?? ??
springdoc.default-consumes-media-type=application/json
springdoc.default-produces-media-type=application/json

# API ?? ?? ??
springdoc.paths-to-match=/api/**
```

```
# Swagger UI ??
springdoc.swagger-ui.disable-swagger-default-url=true
springdoc.swagger-ui.display-request-duration=true
springdoc.swagger-ui.tags-sorter=alpha

# API ?? URL ??
springdoc.swagger-ui.url=/api/v3/api-docs
springdoc.swagger-ui.config-url=/api/v3/api-docs/swagger-config


elasticsearch.query.view-count={\"size\":10,\"query\":{\"bool\":{\"must\":[{\"range\":{\'
elasticsearch.query.view-id={\"size\":10,\"query\":{\"bool\":{\"must\":[{\"range\":{\"@ti

elasticsearch.url=http://k11d205.p.ssafy.io:9200
elasticsearch.username=elastic
elasticsearch.password=elastic_kiwi
```

- application-dev.properties

```
#redis
spring.data.redis.host=k11d205.p.ssafy.io
spring.data.redis.port=6379
spring.data.redis.password=wiki
#endpoint
management.endpoints.web.exposure.include=health,info
management.endpoint.health.show-details=always
```

- jenkions_kiwi_backend.env

```
DOCKERHUB_CREDENTIALS=docker_hub_token
DOCKER_REPO=limnyn/crewin
DOCKER_TAG=kiwi_backend
DOCKER_IMAGE=limnyn/crewin:kiwi_backend
DOCKER_NETWORK=kiwi_network
BLUE_PORT=8021
GREEN_PORT=8022
CONTAINER_NAME_BLUE=kiwi_springboot_server_blue
CONTAINER_NAME_GREEN=kiwi_springboot_server_green
CONTAINER_URL=K11d205.p.ssafy.io
NGINX_CONTAINER_NAME=nginx
NGINX_CONFIG_PATH=/home/ubuntu/dev/nginx/conf.d/default.conf
DEPLOY_LOG_PATH=/home/ubuntu/dev/springboot/deploy.log
MATTERMOST_CHANNEL_NAME=d205-mr-alram
MATTERMOST_WEBHOOK_URL=https://meeting.ssafy.com/hooks/8b16opypkpnh7yizpcyurkg7cw
```

3) 배포 시 특이사항 기재

- 안드로이드 App 및 apk가 4종류입니다.
  1. Kiosk

     키오스크 메인 App
  2. Manager

     점주 전용 관리자 App
  3. Payment Terminal

     결제 단말 App
  4. Payment Receiver

사용자 결제 App

4) DB 접속 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- 2번의 application.properties참고