Gitlab 소스 클론 이후 빌드 및 배포 정리 문서

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

Back-End

IDE	2024.1
Java	17
Spring Boot	3.3.1
Redis	7.2.5
MySql	8.0.38
Nginx	1.27.0
Docker	27.0.3
Jenkins	2.469

Front-End

IDE	1.92.1
TypeScript	5.2.2
React	18.2.0
Node	20.12.2
Npm	10.5.0
Tailwindcss	3.4.6

2) 빌드 시 사용되는 환경 변수 내용 상세 기재

Back-End

- 파이프라인 파일

▼ /backend/Jeniknsfile

```
pipeline {
    agent any
    environment {
        DOCKERHUB_CREDENTIALS = credentials('docker_hub_to
        CREWIN_ENV = credentials('jenkins_crewin_backend.e
    }
    stages {
        stage('Load Environment Variables') {
            steps {
                script {
                    // jenkins_crewin_backend.env 파일 내용을
                    def props = readProperties file: "${CR
                    env.DOCKER_REPO = props.DOCKER_REPO
                    env.DOCKER_TAG = props.DOCKER_TAG
                    env.DOCKER_IMAGE = "${props.DOCKER_REP
                    env.BLUE_PORT = props.BLUE_PORT
                    env.GREEN_PORT = props.GREEN_PORT
                    env.CONTAINER_NAME_BLUE = props.CONTAI
                    env.CONTAINER_NAME_GREEN = props.CONTA
                    env.CONTAINER_URL = props.CONTAINER_UR
                    env.NGINX_CONTAINER_NAME = props.NGINX_
                    env.NGINX_CONFIG_PATH = props.NGINX_CO
                    env.MATTERMOST_CHANNEL_NAME = props.MA
                    env.MATTERMOST_WEBHOOK_URL = props.MAT
                    env.DOCKER_NETWORK = props.DOCKER_NETW
                    env.DEPLOY_LOG_PATH = props.DEPLOY_LOG_
                }
            }
        }
        stage('Notify Build Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git show -
                    def Author_Name = sh(script: "git show
                    def Commit_Message = sh(script: "git l
```

```
                // 모든 커밋 메시지 가져오기
                def allCommits = sh(script: "git log -
                def formattedCommits = allCommits.spli
                    def escapedLine = line.replaceAll(
                    "• ${escapedLine}"
                }.join('\\n')
                def message = """
                    #### 🌐BE 빌드 시작
                    **빌드 번호** $env.JOB_NAME #$env.BU
                    **브랜치:** $env.GIT_BRANCH
                    **작성자:** $Author_ID ($Author_Name
                    **빌드 URL:** [Details]($env.BUILD_
                    **포함된 커밋:**
                    $formattedCommits
                """.stripIndent()

                mattermostSend(
                    color: '#439FE0',
                    message: message,
                    endpoint: "$env.MATTERMOST_WEBHOOK_
                    channel: "$env.MATTERMOST_CHANNEL_
                    icon: 'https://jenkins.io/images/l
                )
            }
        }
    }
    stage('Checkout') {
        steps {
            checkout scm
        }
    }
    stage('Prepare Config') {
        steps {
            withCredentials([file(credentialsId: 'crew
                sh '''
                    mkdir -p backend/src/main/resource
                    chmod -R 755 backend/src/main/reso
```

```
                    cp $CONFIG_FILE backend/src/main/r
                '''
            }
        }
    }
    stage('Build & Test') {
        steps {
            sh 'cd backend && chmod +x gradlew && ./gr
        }
    }
    stage('Build Docker Image') {
        steps {
            script {
                sh 'cd backend && docker build -t $DOC
            }
        }
    }
    stage('Login to Docker Hub') {
        steps {
            script {
                withCredentials([usernamePassword(cred
                    sh 'docker login -u $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: 'Gumi-Server',
                        transfers: [
                            sshTransfer(
                                execCommand: """
                                    set -x  # 디버깅
                                    exec > >(tee $L
                                    docker pull $DO
                                    container_id=\$
                                    echo "컨테이너 I
                                    if [[ \$(docke
                                        NEW_PORT=$(
                                        NEW_NAME=$(
                                        OLD_NAME=$(
                                        CURRENT_POR
                                    else
                                        NEW_PORT=$(
                                        NEW_NAME=$(
                                        OLD_NAME=$(
                                        CURRENT_POR
                                    fi
                                    docker run -d

                                    sleep 10
                                    if curl -sf -m
                                        # Update Ng
                                        sed -i "s/
                                        sed -i "s/
                                        # Restart
                                        docker res
                                        # Stop and
                                        docker sto
                                        docker rm
                                        echo "Swit
                                        exit 0
                                    else
                                        docker sto
                                        docker rm
```

```
                                                    echo "Depl
                                                    exit 1
                                                fi
                                            """
                                        )
                                    ]
                                )
                            ]
                        )
                    }
                }
            }

    }
    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --
                def Author_Name = sh(script: "git show -s
                def Commit_Message = sh(script: "git log -
                def Build_Status = currentBuild.result ?:
                def Status_Color = Build_Status == 'SUCCES
                def Status_Text = Build_Status == 'SUCCESS

                def allCommits = sh(script: "git log --pre
                def formattedCommits = allCommits.split('\
                    def escapedLine = line.replaceAll("([\
                    "• ${escapedLine}"
                }.join('\\n')

                def message = """
                    #### 🌐BE $Status_Text
                    **빌드 번호** $env.JOB_NAME #$env.BUILD_
                    **브랜치:** $env.GIT_BRANCH
                    **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
                )
            }
        }
    }
}
```

1. **Load Environment Variables (환경 변수 로드)**
   - `jenkins_crewin_backend.env` 파일에서 파이프라인에 필요한 환경 변수를 읽어와서 Jenkins 환경 변수로 설정합니다. 이 변수들은 이후 단계에서 사용됩니다.
2. **Notify Build Start (빌드 시작 알림)**
   - Git에서 마지막 커밋 정보를 가져와 Mattermost로 빌드 시작을 알리는 메시지를 전송합니다. 메시지에는 작성자, 브랜치, 커밋 내역 등이 포함됩니다.
3. **Checkout**
   - 소스 코드를 Git에서 체크아웃합니다.
4. **Prepare Config (설정 준비)**
   - `application.yml` 설정 파일을 백엔드 프로젝트의 리소스 디렉토리에 복사합니다.
5. **Build & Test (빌드 및 테스트)**
   - Gradle을 사용하여 프로젝트를 빌드하고 테스트를 실행합니다.
6. **Build Docker Image (Docker 이미지 빌드)**
   - 빌드된 애플리케이션을 기반으로 Docker 이미지를 생성합니다.
7. **Login to Docker Hub (Docker Hub 로그인)**
   - Docker Hub에 로그인하여 Docker 이미지를 푸시할 준비를 합니다.
8. **Push Docker Image (Docker 이미지 푸시)**
   - 생성된 Docker 이미지를 Docker Hub에 푸시합니다.

9. **Deploy and Update Nginx (배포 및 Nginx 업데이트)**
    - 원격 서버에 SSH로 접속하여 Blue-Green 배포를 수행합니다.
    - 새로운 컨테이너를 실행하고, Nginx 설정을 업데이트하며, 새로운 버전이 정상적으로 동작하는지 확인한 후 이전 버전을 종료합니다.
10. **Post (결과 알림)**
    - 빌드가 완료된 후 성공 여부에 따라 Mattermost로 결과를 알립니다. 결과 메시지에는 빌드 상태, 작성자, 커밋 내역 등이 포함됩니다.

Front-End

- 파이프라인 파일

▼ **/frontend/Jeniknsfile**

```
pipeline {
    agent any
    tools {
        nodejs 'node 20.2.0'
    }
    environment {
        DOCKERHUB_CREDENTIALS = credentials('docker_hub_to
        CREWIN_ENV = credentials('jenkins_crewin_frontend.
    }
    stages {
        stage('Load Environment Variables') {
            steps {
                script {
                    // jenkins_crewin_backend.env 파일 내용을
                    def props = readProperties file: "${CR
                    env.MATTERMOST_CHANNEL_NAME = props.MA
                    env.MATTERMOST_WEBHOOK_URL = props.MAT
                    env.DOCKER_CONTAINER_NAME = props.DOCK
                    env.DIST_PATH = props.DIST_PATH
                    env.REMOVE_PREFIX = props.REMOVE_PREFI
                    env.HOST_PATH = props.HOST_PATH
                }
```

```
            }
        }
        stage('Notify Build Start') {
            steps {
                script {
                    def Author_ID = sh(script: "git show -
                    def Author_Name = sh(script: "git show
                    def Commit_Message = sh(script: "git l

                    def allCommits = sh(script: "git log -
                    def formattedCommits = allCommits.spli
                        def escapedLine = line.replaceAll(
                        "• ${escapedLine}"
                    }.join('\\n')

                    def message = """
                        ####  🧑‍💻FE 빌드 시작
                        **빌드 번호** $env.JOB_NAME #$env.BU
                        **브랜치:** $env.GIT_BRANCH
                        **작성자:** ${Author_ID} (${Author_N
                        **빌드 URL:** [Details]($env.BUILD_
                        **포함된 커밋:**
                        $formattedCommits
                    """.stripIndent()

                    mattermostSend(
                        color: '#439FE0',
                        message: message,
                        endpoint: "$env.MATTERMOST_WEBHOOK_
                        channel: "$env.MATTERMOST_CHANNEL_
                        icon: 'https://jenkins.io/images/l
                    )
                }
            }
        }
        stage('Checkout') {
            steps {
                checkout scm
```

```
            }
        }
        stage('Prepare Config') {
            steps {
                withCredentials([file(credentialsId: 'crew
                    sh '''
                        chmod -R 755 frontend
                        cp $CONFIG_FILE frontend/.env
                    '''
                }
            }
        }
        stage('Build') {
            steps {
                script {
                    sh '''
                        cd frontend
                        rm package-lock.json
                        npm install rollup @rollup/plugin-
                        npm install esbuild
                        npm install
                        npm run build
                        chmod -R 755 dist
                        ls -al dist
                    '''
                }
            }
        }
        stage('Transfer') {
            steps {
                script {
                    sh '''
                        cd frontend/dist
                        ls -la
                    ''' // 파일이 빌드 디렉토리에 존재하는지 확인
                }
                sshPublisher(
                    publishers: [
```

```
                        sshPublisherDesc(
                            configName: 'Gumi-Server',
                            transfers: [
                                sshTransfer(
                                    sourceFiles: "${DIST_PA
                                    removePrefix: "${REMOVE
                                    remoteDirectory: "${HO
                                )
                            ],
                            usePromotionTimestamp: false,
                            alwaysPublishFromMaster: false
                        )
                    ]
                )
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh "docker restart ${DOCKER_CONTAINER_
                }
            }
        }
    }

    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --
                def Author_Name = sh(script: "git show -s
                def Commit_Message = sh(script: "git log -
                def Build_Status = currentBuild.result ?:
                def Status_Color = Build_Status == 'SUCCES
                def Status_Text = Build_Status == 'SUCCESS

                def allCommits = sh(script: "git log --pre
                def formattedCommits = allCommits.split('\
                    def escapedLine = line.replaceAll("([\
```

```
                    "• ${escapedLine}"
                }.join('\\n')

                def message = """
                    ####  👨‍💻FE $Status_Text
                    **빌드 번호** $env.JOB_NAME #$env.BUILD_
                    **브랜치:** $env.GIT_BRANCH
                    **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
                )
            }
        }
    }
}
```

1. **Load Environment Variables (환경 변수 로드)**
   - `jenkins_crewin_frontend.env` 파일에서 파이프라인에서 사용할 환경 변수를 읽어와 Jenkins 환경 변수로 설정합니다. 이후 단계에서 이 변수들을 사용합니다.
2. **Notify Build Start (빌드 시작 알림)**
   - Git에서 마지막 커밋 정보를 가져와 Mattermost로 빌드 시작을 알리는 메시지를 전송합니다. 메시지에는 작성자, 브랜치, 커밋 내역 등이 포함됩니다.
3. **Checkout**
   - 소스 코드를 Git에서 체크아웃합니다.
4. **Prepare Config (설정 준비)**

- `crewin-front-env` 파일을 리액트 프로젝트의 `.env` 파일로 복사하여 환경 설정을 준비합니다.

5. **Build**
   - 프로젝트 빌드를 위해 필요한 npm 패키지와 Rollup, esbuild 같은 도구들을 설치한 후, `npm run build` 명령어를 통해 리액트 프로젝트를 빌드합니다. 빌드된 결과물은 `dist` 디렉토리에 저장됩니다.
6. **Transfer (파일 전송)**
   - 빌드된 `dist` 폴더 내의 파일들을 SSH를 통해 원격 서버의 지정된 디렉토리로 전송합니다.
7. **Deploy (배포)**
   - 빌드파일 전송 이후 배포중인 nginx 컨테이너를 재시작합니다. 이를 통해 최신 버전의 리액트 앱이 반영됩니다.
8. **Post (결과 알림)**
   - 빌드가 완료된 후 성공 여부에 따라 Mattermost로 빌드 결과를 알립니다. 메시지에는 빌드 상태, 작성자, 커밋 내역 등이 포함됩니다.

3) 배포 시 특이사항 기재

- EC2 사용하지 않고 우분투 서버 직접 구축해서 배포 및 관리
- 가비아를 통한 도메인 구입 및 인증서 연결
- 백엔드 컨테이너 블루-그린 무중단 배포 구현
- 파이프라인 시작-종료 시 빌드 결과 및 상태 알림 Mattermost 채널에 전송함으로써 빌드 관리

  ▼ Mattermost 이미지 첨부


D204_CREW-IN
6
3
서버 확인하는법
빌드 번호 crewin_react_cicd #209
브랜치: origin/FE/develop
작성자: 박준식 (akfldkskgorn@gmail.com)
빌드 URL: Details
포함된 커밋:
• 7df2688 - fix : location error 페이지 추가 (박준식)
이문현[구미_2반_D204]팀원 BOT 오후 5:03
FE 빌드 성공
빌드 번호 crewin_react_cicd #209
브랜치: origin/FE/develop
작성자: 박준식 (akfldkskgorn@gmail.com)
빌드 URL: Details
포함된 커밋:
• 7df2688 - fix : location error 페이지 추가 (박준식)
이문현[구미_2반_D204]팀원 BOT 오후 5:16
BE 빌드 시작
빌드 번호 crewin_springboot_cicd #287
브랜치: origin/BE/develop
작성자: 이문현 (hogandi001@gmail.com)
빌드 URL: Details
포함된 커밋:
• f06bff0 - fix : 유저 검증 쿼리 수정 (이문현)
이문현[구미_2반_D204]팀원 BOT 오후 5:17
BE 빌드 성공
빌드 번호 crewin_springboot_cicd #287
브랜치: origin/BE/develop
작성자: 이문현 (hogandi001@gmail.com)
빌드 URL: Details
포함된 커밋:
• f06bff0 - fix : 유저 검증 쿼리 수정 (이문현)
D204_CREW-IN에 글쓰기

- 파이프라인 : Gitlab webhook과 연동해 BE/develop 브랜치, FE/develop 브랜치 push 이벤트 발생 시 수행

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

Back-End

▼ application.yml

```
spring:
  application:
    name: crewin

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://210.105.105.200:3306/crewin?useSSL=f
    username: root
    password: crewin1234

  data:
    redis:
      port: 16379
      host: 210.105.105.200
      password: crewin1234

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        jdbc:
          time_zone: Asia/Seoul
        #        show_sql: true # 모든 로그 출력은 가급적 로거를
        format_sql: true
        default_batch_fetch_size: 50
    open-in-view: false

  mail:
```

```
    host: smtp.gmail.com
    port: 587
    username: crewin888@gmail.com
    password: ganh cwmh wrun weat

logging.level:
  org.hibernate.SQL: debug
  org.hibernate.orm.jdbc.bind: trace

cloud:
  aws:
    region:
      static: ap-northeast-2
    credentials:
      access-key: AKIA3FLD2JBBAM4JQNF2
      secret-key: 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
    s3:
      bucket: crewin-bucket
    stack:
      auto: false

image:
  prefix: https://crewin-bucket.s3.ap-northeast-2.amazonaw
  folder: crewin/
  default:
    session-poster: https://crewin-bucket.s3.ap-northeast-
    member-profile: https://crewin-bucket.s3.ap-northeast-
    crew-main-logo: https://crewin-bucket.s3.ap-northeast-
    crew-sub-logo: https://crewin-bucket.s3.ap-northeast-2
    crew-banner: https://crewin-bucket.s3.ap-northeast-2.a
security:
  secret-key: 5E7iGhZfEHuzHFicUyhubeDWicXbv/w40V9CttHAC+0l

frontend:
  server:
    url: https://crew-in.site

logging:
```

```
  level:
    org.springframework.web.servlet.mvc.method.annotation.

#server:
#  ssl:
#    key-store: classpath:keystore.p12
#    key-store-type: PKCS12
#    key-store-password: crewin1234
```

▼ jenkins_crewin_backend.env 파이프라인 환경 변수

```
DOCKERHUB_CREDENTIALS=docker_hub_token
DOCKER_REPO=limnyn/crewin
DOCKER_TAG=backend
DOCKER_IMAGE=${DOCKER_REPO}:${DOCKER_TAG}
DOCKER_NETWORK=crewin_network
BLUE_PORT=12380
GREEN_PORT=12381
CONTAINER_NAME_BLUE=crewin_springboot_server_blue
CONTAINER_NAME_GREEN=crewin_springboot_server_green
CONTAINER_URL=210.105.105.200
NGINX_CONTAINER_NAME=crewin_nginx
NGINX_CONFIG_PATH=/home/limnyn/dev/crewin/nginx/conf.d/def
DEPLOY_LOG_PATH=/home/limnyn/dev/crewin/springboot/deploy.
MATTERMOST_CHANNEL_NAME=D204_CREW-IN
MATTERMOST_WEBHOOK_URL=https://meeting.ssafy.com/hooks/pb3
```

Front-End

▼ .env

```
VITE_SERVER_URL="https://crew-in.site/api"
VITE_PROXY_URL="https://crew-in.site"

VITE_NAVER_MAPS_API_KEY="o41u7sh7bd"
VITE_TMAP_API_KEY="Kgg7KvuInljsY2bjRMsi2I3WSeGwdgh6ZrRd7qn
```

▼ jenkins_crewin_frontend.env 파이프라인 환경 변수

```
DOCKER_CONTAINER_NAME=crewin_nginx
DIST_PATH=frontend/dist/**/*
REMOVE_PREFIX=frontend/dist
HOST_PATH=dev/crewin/react/build/dist
MATTERMOST_CHANNEL_NAME=D204_CREW-IN
MATTERMOST_WEBHOOK_URL=https://meeting.ssafy.com/hooks/pb3
```