

포팅 매뉴얼

1. version

Backend

Java 11

Spring Boot 2.7.16

Frontend

Node.js 18.17.0

React 18.2.0

DB

MySQL 8.1.0

Redis 7.2.2

MongoDB 7.0.2

IDE

IntelliJ 2023.1.5

Gradle 8.3

VSCode 1.84.2

CI/CD

Ubuntu 20.04 LTS

Docker 24.0.6

Jenkins 2.414.3

2. Ubuntu 환경 설정

기본 설치

```
sudo apt-get update
sudo apt-get install openjdk-11-jdk
```

Jenkins 설치

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins
```

key확인 후

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys {key}
sudo apt install jenkins
sudo systemctl status jenkins
```

Docker 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo systemctl status docker
```

DB 설치

```
sudo docker pull mysql
sudo docker run -d -p 3306:3306 --name 컨테이너이름 -e MYSQL_ROOT_PASSWORD=루트비밀번호 mysql
sudo docker pull mongo
sudo docker run --name 컨테이너이름 -e MONGO_INITDB_ROOT_USERNAME=이름 -e MONGO_INITDB_ROOT_PASSWORD=비밀번호 \
-v ~/mongo:/data/db -d -p 27017:27017 mongo
sudo docker pull redis
sudo docker run -d -p 6379:6379 --name 컨테이너이름 redis
```

SSL 설치

```
sudo snap install certbot --classic
sudo certbot certonly --nginx
sudo cp -r letsencrypt /var/lib/jenkins/workspace/Almighty/frontend/blog
```

3. 젠킨스 환경 설정

Pipeline script

```
pipeline {
    agent any
    tools {

        // If Jenkins has a Gradle installation, you can specify it here
        gradle 'Gradle' // Ensure you have a Gradle tool with this name or adjust accordingly
    }

    stages {
        stage('gitlab clone') {
            steps {
                git branch: 'release', // Assuming 'develop_backend' is your backend branch
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s09-final/S09P31S103.git'
            }
        }
        stage('Back build') {
            steps {
                dir('backend/A201/'){ // Adjust the directory path accordingly
                    sh 'sudo sed -i s%{db_url}%${DB_URL}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{db_id}%${DB_ID}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{db_pw}%${DB_PW}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{db_name}%${DB_NAME}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{S3_KEY}%${S3_KEY}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{BMS_URL}%${BMS_URL}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{MAIL}%${MAIL}%g ./src/main/resources/application.yml'
                    sh 'sudo sed -i s%{MAIL_PW}%${MAIL_PW}%g ./src/main/resources/application.yml'
                    sh 'sudo chmod +x gradlew'
                    sh 'sudo ./gradlew clean build'
                    sh 'sudo docker build -t back:1.0 .'
                }
            }
        }
    }
}
```

```

    }
    stage('bms build') {
        steps {
            dir('bms'){ // Adjust the directory path accordingly
                sh 'sudo sed -i s/${db_url}/${DB_URL}%g ./src/main/resources/application.yml'
                sh 'sudo sed -i s/${db_id}/${DB_ID}/g ./src/main/resources/application.yml'
                sh 'sudo sed -i s/${db_pw}/${DB_PW}/g ./src/main/resources/application.yml'
                sh 'sudo sed -i s/${db_name}/${DB_NAME}/g ./src/main/resources/application.yml'
                sh 'sudo chmod +x gradlew'
                sh 'sudo ./gradlew clean build'
                sh 'sudo docker build -t bms:1.0 .'
            }
        }
    }
    stage('Front build') {
        steps {
            dir('frontend/blog/'){ // Adjust the directory path accordingly
                sh 'sudo docker build -t front:1.0 .'
            }
        }
    }
    stage('Docker start') {
        steps {
            script {
                def containerName = 'back' // 여기에 검사할 도커 컨테이너의 이름
                def isContainerRunning = sh(script: "sudo docker ps --filter name=${containerName} --format '{{.ID}}'",
                    returnStatus: true) == 0

                if (isContainerRunning) {
                    echo "도커 컨테이너 '${containerName}'가 실행 중입니다. 파이프라인을 종료합니다."
                    sh "sudo docker stop ${containerName}"
                }
                sh "sudo docker rm ${containerName}"
                sh "sudo docker run -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -d -p 8080:8080 --name back back:1.0"
            }
            script {
                def containerName = 'bms' // 여기에 검사할 도커 컨테이너의 이름
                def isContainerRunning = sh(script: "sudo docker ps --filter name=${containerName} --format '{{.ID}}'",
                    returnStatus: true) == 0

                if (isContainerRunning) {
                    echo "도커 컨테이너 '${containerName}'가 실행 중입니다. 파이프라인을 종료합니다."
                    sh "sudo docker stop ${containerName}"
                }
                sh "sudo docker rm ${containerName}"
                sh "sudo docker run -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -d -p 8085:8085 --name bms bms:1.0"
            }
            script {
                def containerName = 'front' // 여기에 검사할 도커 컨테이너의 이름
                def isContainerRunning = sh(script: "sudo docker ps --filter name=${containerName} --format '{{.ID}}'",
                    returnStatus: true) == 0

                if (isContainerRunning) {
                    echo "도커 컨테이너 '${containerName}'가 실행 중입니다. 파이프라인을 종료합니다."
                    sh "sudo docker stop ${containerName}"
                }
                sh "sudo docker rm ${containerName}"
                sh "sudo docker run -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -d -p 80:80 -p 443:443 --name front front:1.0"
            }
            script {
                def isContainerRunning = sh(script: "sudo docker images -f 'dangling=true' -q", returnStatus: true) == 0

                if (isContainerRunning) {
                    echo "사용하지 않는 컨테이너 삭제"
                    sh "sudo docker rmi -f \$(sudo docker images -f 'dangling=true' -q)"
                }
            }
        }
    }
}
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/p15zf1ksy78rje1hgmb99tok8c',
            )
        }
    }
}

```


```
channel: 'A201_notice'
    )
}
}
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/p15zf1ksy78rje1hgmb99tok8c',
            channel: 'A201_notice'
        )
    }
}
}
```

4. 외부 서비스

- Chat GPT API

OpenAI Platform

Explore developer resources, tutorials, API docs, and dynamic examples to get the most out of OpenAI's platform.

 <https://platform.openai.com/docs/overview>



- Amazon S3

Amazon S3란 무엇인가요? - Amazon Simple Storage Service

Amazon Simple Storage Service(Amazon S3)를 사용하여 언제 어디서든 원하는 양의 데이터를 저장하고 검색하는 방법을 알아봅니다. 본 설명서는 버킷, 객체 및 관련 구성 등 Amazon S3 개념들을 설명하며 일반적인 작업에 대한 코드 예제를 포함합니다.

 https://docs.aws.amazon.com/ko_kr/AmazonS3/latest/userguide/Welcome.html

