포팅 메뉴얼

1. 개발 환경

1.1 BackEnd

(1) JAVA

- JDK 17
- spring-boot 3.4.3
- spring-dependency-management 1.1.7
- gradle 8.13

1.2 DataBase

- MYSQL 8.0.41
- Redis 7.4.2

1.3 Server / Infra

- Ubuntu 22.04.4 LTS
- Jenkins 2.426
- Docker 28.0.1

1.4 Android

- Kotlin 2.1.10
- Gradle Version 8.9
- Android Gradle Plugin Version 8.7.3
- Web3j 4.12.3

- google.zxing 3.5.2

1.5 BlockChain

- hardHat 2.22.19
- Solidity 0.8.28
- ethers 5.7.2
- openzeppelin 5.2.0
- dotenv 16.4.7

1.5 IDE

- IntelliJ IDEA Community Edition 2024.3.2.2
- Visual Studio Code 1.99.0

2. 빌드시 사용되는 환경변수

2.0 EC2 인스턴스에서 Jenkins 컨테이너 빌드 명령어

```
sudo docker run -d -p 9090:8080 \
-v /var/run/docker.sock:/var/run/docker.sock \
-v jenkins_home:/var/jenkins_home \
--restart=always \
--name jenkins \
jenkins/jenkins:lts
```

2.1 파이프라인 파일 (환경변수)

```
pipeline {
    agent any
```

```
environment {
    DOCKER_IMAGE =  'nickjo0126/spring'
    DOCKER_CONTAINER = credentials('DOCKER_CONTAINER')
    DOCKER_PORT = credentials('DOCKER_PORT')
    DOCKER_PATH = '/home/ubuntu/backend-docker' // EC2 서버의 Docker 작

    EC2_USER = credentials('EC2_USER')
    EC2_IP = credentials('EC2_IP')
    SSH_KEY = credentials('SSH_KEY')

    SPRING_PROFILES_ACTIVE = 'dev'
    DB_URL = credentials('DB_URL')
    DB_USERNAME = credentials('DB_USERNAME')
    DB_PASSWORD = credentials('DB_PASSWORD')

    JWT_SECRET = credentials('JWT_SECRET')
    AES_256 = credentials('AES_256')
    REDIS = credentials('REDIS')

    SPRING_DATA_REDIS_HOST = credentials('REDIS')
    SPRING_DATA_REDIS_PORT = '6379'
    REDIS_PASSWORD = credentials('REDIS_PASSWORD')

    BC_PRIVATE_KEY_1 = credentials('BC_PRIVATE_KEY_1')
    BC_PRIVATE_KEY_2 = credentials('BC_PRIVATE_KEY_2')
    BC_PRIVATE_KEY_3 = credentials('BC_PRIVATE_KEY_3')
    BC_PRIVATE_KEY_4 = credentials('BC_PRIVATE_KEY_4')
    BC_PRIVATE_KEY_5 = credentials('BC_PRIVATE_KEY_5')
    BC_PRIVATE_KEY_6 = credentials('BC_PRIVATE_KEY_6')
    BC_FORWARDER = credentials('BC_FORWARDER')
    BC_TOKEN = credentials('BC_TOKEN')
    BC_LECTURE = credentials('BC_LECTURE')

    JAVA_HOME = '/opt/java/openjdk'
    GRADLE_HOME = '/opt/gradle/gradle-8.13'
    PATH = "${JAVA_HOME}/bin:${GRADLE_HOME}/bin:${env.PATH}"
}
```

```
tools {
    jdk 'JDK17'
    gradle 'Gradle 8.13'
}

stages {
    stage('Clone Repository') {
        steps {
            echo 'Cloning the repository...'
            git branch: 'BE/dev',
                url: 'https://lab.ssafy.com/s12-blockchain-nft-sub1/S12P21D210.git
                credentialsId: 'GitLab-PAT'
        }
    }
    stage('Build Application') {
        steps {
            echo 'Building the application with Gradle Wrapper...'
            dir('Backend') {
                sh 'gradle clean build'
                sh 'ls -al $(pwd)/build/libs'
            }
        }
    }
    stage('Build Docker Image') {
        steps {
            echo 'Building the Docker image...'
            dir('Backend') {
                sh 'cp build/libs/backend-0.0.1-SNAPSHOT.jar .'
                sh 'docker build -t ${DOCKER_IMAGE}:latest .'
            }
        }
    }
    stage('Save and Transfer Docker Image') {
        steps {
            echo 'Saving and transferring Docker image to EC2...'
            sh """
            docker save ${DOCKER_IMAGE}:latest | gzip > backend-0.0.1-SNAP
```

```
            """
            sshPublisher(publishers: [
                sshPublisherDesc(
                    configName: 'EC2-Server',
                    transfers: [
                        sshTransfer(
                            sourceFiles: 'backend-0.0.1-SNAPSHOT.tar.gz'
                        )
                    ]
                )
            ])
        }
    }
    stage('Deploy to EC2') {
        steps {
            echo 'Deploying the application on EC2...'
            sshPublisher(publishers: [
                sshPublisherDesc(
                    configName: 'EC2-Server',
                    transfers: [
                        sshTransfer(
                            execCommand: """
                                mkdir -p ${DOCKER_PATH}
                                mv backend-0.0.1-SNAPSHOT.tar.gz ${DOCKER_PATH}/
                                docker stop ${DOCKER_CONTAINER} || true
                                docker rm ${DOCKER_CONTAINER} || true
                                docker rmi ${DOCKER_IMAGE}:latest || true
                                docker load < ${DOCKER_PATH}/backend-0.0.1-SNAPSH
                                docker run -d --name ${DOCKER_CONTAINER} \
                                  --network learnauth \
                                  -p ${DOCKER_PORT}:${DOCKER_PORT} \
                                  -e SPRING_PROFILES_ACTIVE=dev \
                                  -e PORT=${DOCKER_PORT} \
                                  -e JWT_SECRET=${JWT_SECRET} \
                                  -e BC_PRIVATE_KEY_1=${BC_PRIVATE_KEY_1} \
                                  -e BC_PRIVATE_KEY_2=${BC_PRIVATE_KEY_2} \
                                  -e BC_PRIVATE_KEY_3=${BC_PRIVATE_KEY_3} \
                                  -e BC_PRIVATE_KEY_4=${BC_PRIVATE_KEY_4} \
```

```
                        -e BC_PRIVATE_KEY_5=${BC_PRIVATE_KEY_5} \
                        -e BC_PRIVATE_KEY_6=${BC_PRIVATE_KEY_6} \
                        -e BC_FORWARDER=${BC_FORWARDER} \
                        -e BC_TOKEN=${BC_TOKEN} \
                        -e BC_LECTURE=${BC_LECTURE} \
                        -e AES_256=${AES_256} \
                        -e SPRING_DATA_REDIS_HOST="${SPRING_DATA_RED
                        -e SPRING_DATA_REDIS_PORT="${SPRING_DATA_RED
                        -e SPRING_DATA_REDIS_PASSWORD="${REDIS_PASSV
                        -e DB_URL="${DB_URL}" \
                        -e DB_USERNAME=${DB_USERNAME} \
                        -e DB_PASSWORD=${DB_PASSWORD} \
                        ${DOCKER_IMAGE}:latest
                    """.stripIndent()
                )
            ]
        )
    ])
  }
 }
 }

 post {
  always {
   echo 'Cleaning workspace...'
   cleanWs()
  }
  success {
   echo 'Deployment successful!'
  }
  failure {
   echo 'Deployment failed.'
  }
 }
}
```

2.2 Docker 파일

- Spring- Dockerfile 경로
  - S12P21D210/Backend

2.3 사용 포트 번호

- Spring
  - 8080
- jenkins
  - 9000
- mysql
  - 3306
- redis
  - 6379
- Node Exporter
  - 9100
- Prometheus
  - 9090
- Grafana
  - 3000/tcp

jenkins 9000
spring-boot 8080

2.4 Android 내 Youtube API Key

```
YOUTUBE_API_KEY="your_api_key"
```

3. 주요 계정 및 프로퍼티 정의 파일 목록

3.1 spring - application.yml

```
server:
  port: ${PORT}
#
spring:
  config:
    import: optional:file:.env[.properties]

  datasource:
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      maximum-pool-size: 30  # 기본값 10에서 증가
      connection-timeout: 60000  # 60초로 증가
      idle-timeout: 600000  # 10분
      max-lifetime: 1800000  # 30분


  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true

  data:
    redis:
      host: ${REDIS}
```

```
    port: 6379
    password: ${REDIS_PASSWORD}
  jwt:
    secret: ${JWT_SECRET}

  encryption:
    key: ${AES_256}

  server:
    forward-headers-strategy: NATIVE
    tomcat:
      remote-ip-header: x-forwarded-for
      protocol-header: x-forwarded-proto

springdoc:
  server:
    url: https://j12d210.p.ssafy.io

logging:
  level:
    root: INFO
    org.springframework.web: DEBUG
    org.hibernate.SQL: debug
    org.hibernate.orm.jdbc.bind: trace
#logging:
#  level:
#    root: DEBUG

blockchain:
  rpc:
    url: https://rpc-amoy.polygon.technology/
    chain-id: 80002
  relayer:
    private-keys: ${BC_PRIVATE_KEY_3},${BC_PRIVATE_KEY_4},${BC_PRIVATE
  forwarder:
    address: ${BC_FORWARDER}
  cat-token:
    address: ${BC_TOKEN}
```

```
  lecture-system:
    address: ${BC_LECTURE}

management:
  endpoints:
    web:
      exposure:
        include: "prometheus,health,info,metrics"
  endpoint:
    health:
      show-details: "always"
  prometheus:
    metrics:
      export:
        enabled: true
```

3.2 BlockChain

```
import { HardhatUserConfig } from "hardhat/config";
import "@nomicfoundation/hardhat-toolbox";
import "dotenv/config";

const config: HardhatUserConfig = {
  solidity: {
    version: "0.8.28",
    settings: {
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
  networks: {
    hardhat: {},
    amoi: {
      url: process.env.AMOY_URL || "https://rpc-amoy.polygon.technology/",
```

```
      accounts: process.env.PRIVATE_KEY ? [process.env.PRIVATE_KEY] : [],
    },
    polygon: {
      url: process.env.POLYGON_URL || "https://polygon-rpc.com",
      accounts: process.env.PRIVATE_KEY ? [process.env.PRIVATE_KEY] : [],
    },
  },
  etherscan: {
    apiKey: process.env.POLYGONSCAN_API_KEY,
  },
  paths: {
    sources: "./contracts",
    tests: "./test",
    cache: "./cache",
    artifacts: "./artifacts",
  },
};

export default config;
```

앱 사용시 유의사항

- wallet의 경우, 같은 계정이라 할지라도, 유지되지 않을 가능성이 있습니다. 따라서, 회원가입을 새로 진행하시는 것을 추천드립니다.

주요 계정

- **User 1**
  - Email: **user1@example.com**
  - Password: 12345678
- **User 2**
  - Email: **user2@example.com**
  - Password: 12345678
- **User 3**

- Email: **user3@example.com**
- Password: 12345678

4. 외부 서비스 정보

4.1 YouTube Player API Reference for iframe Embeds

https://developers.google.com/youtube/iframe_api_reference

4.2 YouTube Data API v3

https://developers.google.com/youtube/v3?hl=ko