포팅 메뉴얼

빌드 및 배포

0. 초기 세팅

1. EC2 접속

```
# sudo ssh -i [key.pem] [접속 계정]@[도메인]
$ sudo ssh -i J12A601T.pem ubuntu@k12a501.p.ssafy.io
```

2. Docker / Nginx / Docker-Compose 설치
3. Jenkins 설치

1. docker-compose 구성

a. redis: `7.4.3`

```
ubuntu@ip-172-26-5-114:~$ docker exec -it moba-redis redis-server --version
Redis server v=7.4.2 sha=00000000:0 malloc=jemalloc-5.3.0 bits=64 build=c8f507d0914a3bf4
```

b. mySQL: `9.3.0`

```
ubuntu@ip-172-26-5-114:~$ docker exec -it moba-mysql mysql --version
mysql  Ver 9.2.0 for Linux on x86_64 (MySQL Community Server - GPL)
```

1. my-backend (배포 파일): `latest`
2. `docker-compose.yml`

```
services:
  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
```

```
    - "80:80"
    - "443:443"
  networks:
    - nginx-net
  depends_on:
    - checkit-server
    - jenkins
    - portainer
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/nginx.conf
    - /etc/letsencrypt/live/p.ssafy.io/fullchain.pem:/etc/letsencrypt/live/p.ssafy
    - /etc/letsencrypt/live/p.ssafy.io/privkey.pem:/etc/letsencrypt/live/p.ssafy.
checkit-server:
  build:
    context: ../service/checkit
    dockerfile: DockerFile
  container_name: checkit-server
  environment:
    - TZ=Asia/Seoul
  depends_on:
    - checkit_mysql
    - checkit_redis
  networks:
    - nginx-net
    - checkit-net
  env_file:
    - /home/ubuntu/service/checkit/.env

checkit_redis:
  image: redis:latest
  container_name: checkit_redis
  networks:
    - checkit-net
  ports:
    - "6379:6379"
checkit_mysql:
  image: mysql:latest
  container_name: checkit_mysql
```

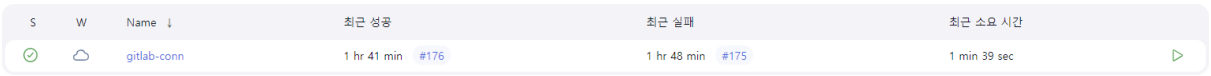
```
    environment:
      MYSQL_ROOT_PASSWORD: "f!!jwpeiourq!@#!@"
      MYSQL_DATABASE: "checkit"
    networks:
      - checkit-net
    ports:
      - "3306:3306"
    healthcheck:
      test: ['CMD', 'mysqladmin', 'ping', '-h', 'localhost', '-u', 'root', '-proot']
      interval: 5s
      timeout: 10s
      retries: 5
    volumes:
      - mysql_checkit_data:/var/lib/mysql
  jenkins:
    build:
      context: ./jenkins
      dockerfile: jenkins.Dockerfile
    container_name: jenkins
    ports:
      - "8080:8080"
    volumes:
      - ./jenkins/jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
    restart: always
    networks:
      - nginx-net
    environment:
      - JENKINS_OPTS=--prefix=/jenkins

  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    command: -H unix:///var/run/docker.sock
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
```

```
    restart: always
    networks:
      - nginx-net

networks:
  nginx-net:
    name: nginx-net
    driver: bridge
  checkit-net:
    name: checkit-net
    driver: bridge

volumes:
  mysql_checkit_data:
  jenkins_home:
  portainer_data:
```

2. Jenkins: 2.492.1

Jenkins 2.492.1

a. 파이프라인 구성



a. 백엔드 pipeline script

```
pipeline {
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s12-final/S12P31A501.git"
        JAR_NAME = "app.jar"
```

```
    BUILD_DIR = "build/libs"
    REMOTE_USER = "ubuntu"
    REMOTE_HOST = "k12a501.p.ssafy.io"
    BRANCH_NAME = "${env.ref ?: 'unknown'}"
  }

  options {
    disableConcurrentBuilds()
  }

  stages {
    stage('Stop if Frontend Branch') {
      steps {
        script {
          echo "💡 Webhook에서 전달받은 브랜치: ${env.BRANCH_NAME
          if (env.BRANCH_NAME == 'fe/dev') {
            echo "🚫 'fe/dev' 브랜치는 백엔드 배포 대상이 아닙니다. 파이프라
            currentBuild.result = 'SUCCESS'
            return
          }
        }
      }
    }

    stage('Clean') {
      steps {
        cleanWs()
      }
    }

    stage('Clone & Checkout') {
      steps {
        script {
          git branch: "${env.BRANCH_NAME}",
            url: "${GIT_URL}",
            credentialsId: 'roots'

          if (env.BRANCH_NAME == "be/dev") {
```

```
                env.SERVICE_NAME = "checkit-server"
                env.CLONE_PATH = "/home/ubuntu/service/checkit"
            }
        }
    }
}

stage('Build') {
    steps {
        sh 'chmod +x ./gradlew'
        sh './gradlew clean bootJar'
    }
}

stage('Send to EC2') {
    steps {
        sshagent(credentials: ['target-server-key']) {
            sh """
                ssh -o StrictHostKeyChecking=no ${REMOTE_USER}@${RE
                scp -o StrictHostKeyChecking=no ${BUILD_DIR}/${JAR_NA
            """
        }
    }
}

stage('Remote Deploy') {
    steps {
        sshagent(credentials: ['target-server-key']) {
            sh """
                ssh -o StrictHostKeyChecking=no ${REMOTE_USER}@${RE
                    cd /home/ubuntu/deploy &&
                    docker-compose build ${SERVICE_NAME} &&
                    docker-compose up -d ${SERVICE_NAME}
                '
            """
        }
    }
}
```

```
    }

    post {
        success {
            echo "✅ ${env.SERVICE_NAME} 배포 성공!"
        }
        failure {
            echo "❌ 배포 실패: 브랜치 ${env.BRANCH_NAME}"
        }
    }
}
```

b. 프론트 pipeline

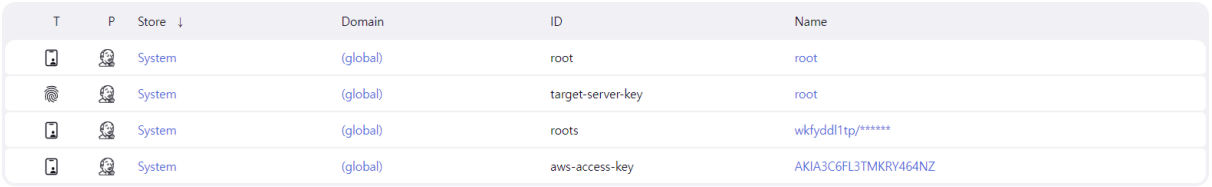
```
pipeline {
    agent any

    tools {
        nodejs "nodejs-24.0.1" // Jenkins에 등록된 Node.js 설치 이름
    }

    environment {
        AWS_REGION = 'ap-northeast-2'        // AWS 리전
        FRONTEND_DIR = 'frontend'            // frontend 디렉토리 이름
        S3_BUCKET = 'checkit-my'             // 배포할 S3 버킷 이름
        DIST_PATH = 'dist'                   // S3 내 경로
        PUSHED_BRANCH = "${env.GIT_BRANCH ?: 'unknown'}" // Git 브랜치
    }

    stages {
        stage('Clone Repository') {
            steps {
                git branch: 'fe/dev', // 필요에 따라 브랜치 수정
                    url: 'https://lab.ssafy.com/s12-final/S12P31A501.git',
                    credentialsId: 'roots'
            }
        }
        stage('Create .env file') {
```

```
        steps {
            dir("${FRONTEND_DIR}") {
                writeFile file: '.env', text: """
VITE_API_BASE_URL=https://k12a501.p.ssafy.io
VITE_REDIRECT_URI_INFO=https://checkit.my
VITE_GITLAB_CLIENT_ID=b7f1549e87321f1d07e5f2dd45ca38debc12fd174
VITE_JIRA_CLIENT_ID=5gepnXtpeNVp2S3V0jArukMeEr7×2dwJ
"""
            }
        }
    }

    stage('Build') {
        steps {
            dir("${FRONTEND_DIR}") {
                sh 'rm -rf node_modules package-lock.json'
                sh 'npm install'
                sh 'npm run build'
            }
        }
    }

    stage('Deploy to S3') {
        steps {
            dir("${FRONTEND_DIR}") {
                withAWS(credentials: 'aws-access-key', region: "${AWS_REGI
                    script {
                        echo "[INFO] 배포할 파일 목록 확인:"
                        sh 'ls -al dist'

                        echo "[INFO] S3로 파일 동기화 시작..."
                        sh "aws s3 sync dist s3://${S3_BUCKET}/${DIST_PATH}

                         sh '''aws cloudfront create-invalidation \
                         --distribution-id E3I57IE4AYPBQ3 \
                         --paths "/*"'''

                        echo "[INFO] ✅ S3 배포 완료!"
```

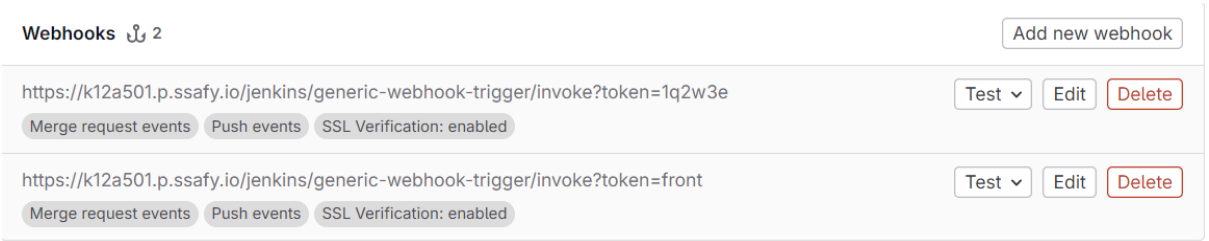
```
                    echo "      👉 https://${S3_BUCKET}.s3.${AWS_REGION}
                }
            }
        }
    }
}
```

c. 수많은 Confidentials

d. Gitlab Webhook 설정



T	P	Store ↓	Domain	ID	Name
		System	(global)	root	root
		System	(global)	target-server-key	root
		System	(global)	roots	wkfyddl1tp/******
		System	(global)	aws-access-key	AKIA3C6FL3TMKRY464NZ



3. AWS 설정

a. SSAFY가 제공한 Nginx 설정 (nginx.conf)

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
```

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}



include    /etc/nginx/mime.types;
default_type application/octet-stream;
client_max_body_size 50M;

sendfile    on;


# SSL 설정 (HTTPS 지원)
ssl_certificate /etc/letsencrypt/live/p.ssafy.io/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/p.ssafy.io/privkey.pem;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;

# HTTP → HTTPS 리디렉션
server {
    listen 80;
    listen [::]:80;
    server_name k12a501.p.ssafy.io;

    location / {
        return 308 https://$host$request_uri;
    }
}

# 메인 HTTPS 서버
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name k12a501.p.ssafy.io;

    # API 요청을 Spring Boot 백엔드로 프록시
```

```
        location /api/ {
            proxy_pass http://checkit-server:8080;
            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_redirect off;
            proxy_intercept_errors off;
        }


        location /ws/ {
            proxy_pass http://checkit-server:8080;
            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 Authorization $http_authorization;
            proxy_redirect off;
            proxy_intercept_errors off;
            proxy_read_timeout 60s;
        }

        location /portainer {
            proxy_pass http://portainer:9000;
            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_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";

            # 이 부분이 중요: /portainer/로 접근 시 내부 경로 정리
```

```
        rewrite ^/portainer(/.*)$ $1 break;
    }

    location /swagger-ui.html {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTION
        add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-R
        add_header 'Access-Control-Expose-Headers' 'Content-Length,Co
        add_header 'X-Content-Type-Options' 'nosniff';

        proxy_pass http://checkit-server:8080/swagger-ui.html;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /swagger-ui/ {
        proxy_pass http://checkit-server:8080/swagger-ui/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /v3/api-docs/ {
        proxy_pass http://checkit-server:8080/v3/api-docs/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
    location = /v3/api-docs {
        proxy_pass http://checkit-server:8080/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-resources/ {
        proxy_pass http://checkit-server:8080/swagger-resources/;
    }
```

```
        location /webjars/ {
            proxy_pass http://checkit-server:8080/webjars/;
        }


        location = /swagger-ui.html {
            proxy_pass http://checkit-server:8080/swagger-ui.html;
        }


        location /jenkins/ {
            sendfile off;
            proxy_pass http://jenkins:8080;
            proxy_http_version 1.1;
            proxy_redirect off;

            # Required for Jenkins websocket agents
            proxy_set_header   Connection       "upgrade";
            proxy_set_header   Upgrade          $http_upgrade;

            proxy_set_header   Host             $http_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_max_temp_file_size 0;

            client_max_body_size       10m;
            client_body_buffer_size    128k;

            proxy_connect_timeout      90;
            proxy_send_timeout         90;
            proxy_read_timeout         90;
           proxy_request_buffering    off;
        }

        location / {
            root /usr/share/nginx/html;
            index index.html;
```

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

📌개발 환경

외부 서비스

- **Oauth (gitlab)**
- **OpenAI API**
- **NAVER SMTP**
- JIRA

협업 도구

- **이슈 관리 : Jira**
- **형상 관리 : GitLab**
- **커뮤니케이션 : MatterMost, Notion**
- **디자인 : Figma**

FrontEnd

🛠️ 개발 환경

본 프로젝트는 **React Native+ TypeScript** 를 기반으로 개발되었습니다.

아래와 같은 환경에서 실행 및 개발이 가능합니다.

📋 기술 스택

기술	버전
React-Native	`0.76.7`
TypeScript	`>=5.3.3`
zustand	`>=5.0.4`
React Router	`>=4.0.19`
Axios	`>=1.8.4`

💻 개발 환경

- **Node.js**: `>=16.x` (LTS 버전 권장)
- **패키지 매니저**: `npm`, `yarn`, 또는 `pnpm` 중 선택 가능 (기본: `npm`)
- **OS 지원**: Windows, macOS, Linux

🚀 프로젝트 실행 방법

1️⃣ 프로젝트 클론

먼저 GitHub 또는 GitLab에서 프로젝트를 클론합니다.

```
git clone https://lab.ssafy.com/s12-final/S12P31A501.git
cd frontend
```

2️⃣ 패키지 설치

프로젝트 실행에 필요한 패키지를 설치합니다.

```
bash
복사편집
# npm 사용 시
npm install
```

3️⃣ 환경 변수 설정

`.env` 파일을 생성하고 필요한 환경 변수를 설정합니다.

```
VITE_API_BASE_URL=http://checkit.my 리다이렉트 URI 환경변수
VITE_REDIRECT_URI_INFO=http://localhost:5173
VITE_GITLAB_CLIENT_ID=b7f1549e87321f1d07e5f2dd45ca38debc12fd174f1
244c857058592d0efe739
VITE_JIRA_CLIENT_ID=5gepnXtpeNVp2S3V0jArukMeEr7×2dwJ
```

4️⃣ 개발 서버 실행

로컬 개발 환경에서 프로젝트를 실행하려면 다음 명령어를 입력하세요.

```
# 프로젝트 빌드
npm run dev
```

🔗 **실행 후:**

개발 서버가 실행되며, `http://localhost:5173` 에서 프로젝트를 확인할 수 있습니다.

📦 프로젝트 빌드

배포용으로 프로젝트를 빌드하려면 다음 명령어를 실행하세요.

```
# 프로젝트 빌드
npm run build
```

BackEnd

- **Gradle : 8.11.1**
- **Redis : 7.4.2**
- **Spring Boot : 3.4.1**
- **Java : 17**
- **IntelliJ : 2024.3.1.1**

application.yml

```yaml
# application.yml
logging:
  level:
    org.springframework.messaging.simp.stomp: DEBUG
    org.springframework.web.socket: DEBUG
    org.springframework.web.socket.messaging: DEBUG

spring:
  config:
    import:
      - classpath:application-redis.yml
      - classpath:application-mysql.yml
      - classpath:application-jwt.yml
      - classpath:application-oauth.yml
      - classpath:application-mail.yml
      - optional:application-project.yml
      - optional:application-ai.yml
      - optional:file:.env[.properties]
  application:
    name: checkit

  servlet:
    multipart:
      max-file-size: 50MB
      max-request-size: 50MB
  jackson:
    time-zone: Asia/Seoul

  web:
    resources:
      add-mappings: false

server:
  forward-headers-strategy: framework
```

```yaml
springdoc:
  swagger-ui:
    path: /swagger-ui.html
  api-docs:
    enabled: true
    path: /v3/api-docs
```

application-ai.yml

```yaml
# AI 주소값

ai:
  server:
    url: ${AI_SERVER_URL}
  llm:
    type: ${LLM_TYPE}
  api:
    key: ${AI_API_KEY}
```

application-jwt.yml

```yaml
# JWT 설정
app:
  jwt:
    secret: ${JWT_SECRET}
    access-expiration: ${JWT_ACCESS_EXPIRATION}
    refresh-expiration: ${JWT_REFRESH_EXPIRATION}
```

application-mail.yml

```yaml
spring:
  mail:
    host: ${MAIL_HOST}
    port: ${MAIL_PORT}
    username: ${MAIL_USERNAME}
    password: ${MAIL_PASSWORD}
    properties:
      mail.smtp.auth: true
```

```
    mail.smtp.starttls.enable: true
    mail.smtp.connectiontimeout: 1000
    mail.debug: true
```

application-jwt.yml

```
jwt:
  secret: ${JWT_SECRET}
  expiration: 3600000 # 1시간
```

application-mysql.yml

```
spring:
  datasource:
    url: ${MYSQL_URL}
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    properties:
      hibernate:
        format_sql: true
        use_sql_comments: true
        jdbc:
          batch_size: 30
    #       show_sql: true
    hibernate:
      ddl-auto: update
```

application-oauth.yml

```
# OAuth 공통 설정
oauth:
  providers:
    github:
      client-id: ${GITHUB_CLIENT_ID}
      client-secret: ${GITHUB_CLIENT_SECRET}
      redirect-uri: ${GITHUB_REDIRECT_URI}
```

```
      scope: ${GITHUB_SCOPE}
      authorization-uri: https://github.com/login/oauth/authorize
      token-uri: https://github.com/login/oauth/access_token
      user-info-uri: https://api.github.com/user
      user-name-attribute: login

    gitlab:
      client-id: ${GITLAB_CLIENT_ID}
      client-secret: ${GITLAB_CLIENT_SECRET}
      redirect-uri: ${GITLAB_REDIRECT_URI}
      scope: ${GITLAB_SCOPE}
      authorization-uri: https://lab.ssafy.com/oauth/authorize
      token-uri: https://lab.ssafy.com/oauth/token
      user-info-uri: https://lab.ssafy.com/api/v4/user
      user-name-attribute: username
      api-uri: https://lab.ssafy.com/api/v4

    jira:
      client-id: ${JIRA_CLIENT_ID}
      client-secret: ${JIRA_CLIENT_SECRET}
      redirect-uri: ${JIRA_REDIRECT_URI}
      scope: ${JIRA_SCOPE}
      authorization-uri: https://auth.atlassian.com/authorize
      token-uri: https://auth.atlassian.com/oauth/token
      user-info-uri: https://api.atlassian.com/oauth/userinfo
      api-uri: https://api.atlassian.com
```

application-project.yml

```
project:
  invite:
    url: ${PROJECT_INVITE_URL}
```

application-redis.yml

```
spring:
  data:
    redis:
```

```
    host: ${REDIS_HOST}
    port: ${REDIS_PORT}
    database: 0
```

bundle.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.5'
    id 'io.spring.dependency-management' version '1.1.7'
    id 'checkstyle'
}

group = 'com.checkmate'
version = '0.0.1-SNAPSHOT'

bootJar {
    archiveFileName = 'app.jar'   // ← 최종 빌드 결과물을 app.jar로 만듦
}

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
```

```
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-clie
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    implementation 'org.springframework.boot:spring-boot-configuration-proce
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-mail'
    implementation 'org.eclipse.jgit:org.eclipse.jgit:6.5.0.202303070854-r'
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'net.lingala.zip4j:zip4j:2.11.5'
}

tasks.named('test') {
    useJUnitPlatform()
}

checkstyle {
    maxWarnings = 0
    configFile = file('config/checkstyle/naver-checkstyle-rules.xml')
    configProperties = ['checkstyle.suppressions.file': 'config/checkstyle/naver
    toolVersion = '10.21.3'
}
```