포팅 메뉴얼(Porting Manual)

빌드 및 배포

0. 초기 세팅

1. EC2 접속

```
# sudo ssh -i [key.pem] [접속 계정]@[도메인]
$ sudo ssh -i J12A601T.pem ubuntu@j12a601.p.ssafy.io
```

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

1. docker-compose 구성

a. redis: `7.4.2`

```
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: `9.2.0`

```
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:
      - moba-server
      - jenkins
    volumes:
      - ./nginx/conf/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/live/j12a601.p.ssafy.io/fullchain.pem:/etc/letsencrypt/live
      - /etc/letsencrypt/live/j12a601.p.ssafy.io/privkey.pem:/etc/letsencrypt/live/
  moba-server:
    build:
      context: ../services/moba
      dockerfile: Dockerfile
    container_name: moba-server
    environment:
      - TZ=Asia/Seoul
    depends_on:
      - moba-mysql
      - moba-redis
      - moba-mongo
    networks:
      - moba-net
      - shared-net
      - nginx-net
    volumes:
      - /home/ubuntu/services/moba/serviceAccountKey.json:/app/serviceAcco
    env_file:
      - /home/ubuntu/services/moba/.env

  moba-mysql:
    image: mysql:latest
    container_name: moba-mysql
    environment:
      MYSQL_ROOT_PASSWORD: "f!!jwpeiourq!@#!@$"
```

```
    MYSQL_DATABASE: "moba"
  networks:
    - moba-net
  ports:
    - "3307:3306"
  healthcheck:
    test: ['CMD', 'mysqladmin', 'ping', '-h', 'localhost', '-u', 'root', '-proot']
    interval: 5s
    timeout: 10s
    retries: 5
  volumes:
    - mysql_moba_data:/var/lib/mysql
    - /home/ubuntu/services/moba/init.sql:/docker-entrypoint-initdb.d/init.sql

moba-redis:
  image: redis:latest
  container_name: moba-redis
  networks:
    - moba-net
  ports:
    - "6379:6379"

moba-mongo:
  image: mongo:latest
  container_name: moba-mongo
  restart: always
  environment:
    MONGO_INITDB_DATABASE: "chatdb"
  networks:
    - moba-net
  ports:
    - "27018:27017"
  volumes:
    - mongo_moba_data:/data/db

moba-bank:
  build:
    context: ../services/moba-bank
```

```
    dockerfile: Dockerfile
  container_name: moba-bank
  depends_on:
    - bank-mysql
  networks:
    - bank-net
    - nginx-net
    - shared-net
  env_file:
    - /home/ubuntu/services/moba-bank/.env

bank-mysql:
  image: mysql:latest
  container_name: bank-mysql
  environment:
    MYSQL_ROOT_PASSWORD: "f!!jwpeiourq!@#!@bank"
    MYSQL_DATABASE: "moba-bank"
  networks:
    - bank-net
  ports:
    - "3308:3306"
  healthcheck:
    test: ['CMD', 'mysqladmin', 'ping', '-h', 'localhost', '-u', 'root', '-proot']
    interval: 5s
    timeout: 10s
    retries: 5
  volumes:
    - mysql_bank_data:/var/lib/mysql

moba-mydata:
  container_name: moba-mydata
  build:
    context: ../services/moba-mydata
  env_file:
    - ../services/moba-mydata/.env
  depends_on:
    - mydata-mongo
  networks:
```

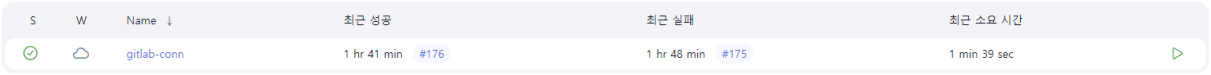
```
      - mydata-net
      - shared-net
      - nginx-net
  mydata-mongo:
    image: mongo:latest
    container_name: mydata-mongo
    volumes:
      - mongo_mydata_data:/data/db
    restart: always
    networks:
      - mydata-net

  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

networks:
  moba-net:
    name: moba-net
    driver: bridge
  nginx-net:
    name: nginx-net
    driver: bridge
  bank-net:
    name: bank-net
    driver: bridge
  shared-net:
```

```
    name: shared-net
    driver: bridge
  mydata-net:
    name: mydata-net
    driver: bridge

volumes:
  mysql_moba_data:
  mongo_moba_data:
  mysql_bank_data:
  mongo_mydata_data:
```

2. Jenkins: 2.492.1

Jenkins 2.492.1

a. 파이프라인 구성



a. 백엔드 pipeline script

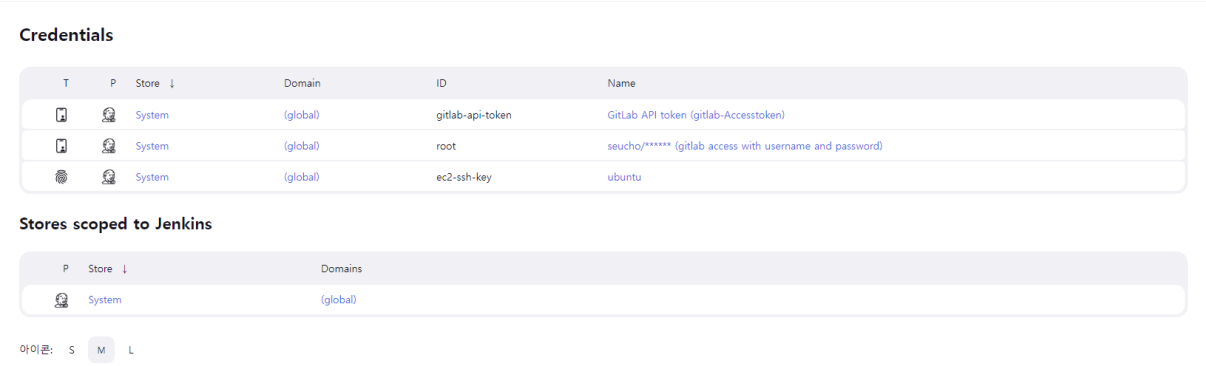
```
pipeline {
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s12-fintech-finance-sub1/S12P21A6
        JAR_NAME = "app.jar"
        BUILD_DIR = "back-end/moba/build/libs"
        REMOTE_USER = "ubuntu"
        REMOTE_HOST = "j12a601.p.ssafy.io"
        BRANCH_NAME = "${env.ref ?: 'unknown'}"
    }
```

```
    options {
        disableConcurrentBuilds()
    }

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

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

                    // 브랜치에 따라 clone 대상 경로 결정
                    if (env.BRANCH_NAME == "BE-moba") {
                        env.SERVICE_NAME = "moba-server"
                        env.CLONE_PATH = "/home/ubuntu/services/moba"
                    } else if (env.BRANCH_NAME == "BE-bank") {
                        env.SERVICE_NAME = "moba-bank"
                        env.CLONE_PATH = "/home/ubuntu/services/moba-bank"
                    } else if (env.BRANCH_NAME == "BE-mydata") {
                        env.SERVICE_NAME = "moba-mydata"
                        env.CLONE_PATH = "/home/ubuntu/services/moba-mydata"
                    } else {
                        error "Unknown branch: ${env.BRANCH_NAME}"
                    }
                }
            }
        }

        stage('Build') {
            steps {
                script {
```

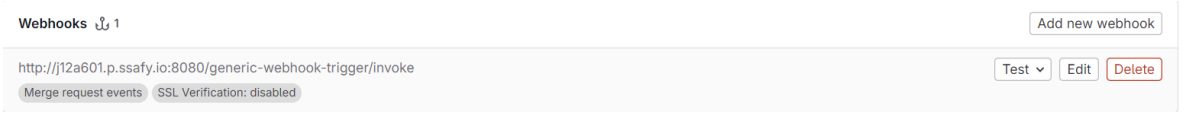
```
            if (env.BRANCH_NAME == "BE-mydata") {
                echo "✅ FastAPI는 별도 빌드 없이 Docker에서 처리됩니다."
            } else {
                dir('back-end/moba') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                }
            }
        }
    }
}

stage('Send to EC2') {
    steps {
        sshagent(credentials: ['ec2-ssh-key']) {
            script {
                if (env.BRANCH_NAME == "BE-mydata") {
                    sh """
                        ssh -o StrictHostKeyChecking=no ${REMOTE_USER}@
                        scp -o StrictHostKeyChecking=no -r ./backend/mydata
                        scp -o StrictHostKeyChecking=no ./backend/requireme
                    """
                } else {
                    sh """
                        ssh -o StrictHostKeyChecking=no ${REMOTE_USER}@
                        scp -o StrictHostKeyChecking=no ${BUILD_DIR}/${JAR
                    """
                }
            }
        }
    }
}

stage('Remote Deploy') {
    steps {
        sshagent(credentials: ['ec2-ssh-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. 수많은 Confidentials



a. Gitlab Webhook 설정



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/j12a601.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j12a601.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 j12a601.p.ssafy.io;

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

# 메인 HTTPS 서버
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name j12a601.p.ssafy.io;

    # API 요청을 Spring Boot 백엔드로 프록시
    location /api/ {
        proxy_pass http://moba-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 /api/banks/ {
                proxy_pass http://moba-bank:8080;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
                proxy_set_header X-Forwarded-Proto $scheme;

                proxy_set_header Authorization $http_authorization;
                proxy_redirect off;
                proxy_intercept_errors off;
            }
```

```
 location /api/mydata/ {
            proxy_pass http://moba-mydata:8000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
            proxy_set_header X-Forwarded-Proto $scheme;

            proxy_set_header Authorization $http_authorization;
            proxy_redirect off;
            proxy_intercept_errors off;
        }




location /api/ws/ {
    proxy_pass http://moba-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;
}


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,Cc
```

```
        add_header 'X-Content-Type-Options' 'nosniff';

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

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

    location /v3/api-docs/ {
        proxy_pass http://moba-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://moba-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://moba-server:8080/swagger-resources/;
    }

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


    location = /swagger-ui.html {
        proxy_pass http://moba-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_fo
            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 (kakao)**
- **OpenAI API**
- **쿨에스엠에스**
- **kakao map**
- **Gmail SMTP**

협업 도구

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

FrontEnd

🛠️ 개발 환경

본 프로젝트는 **expo + 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`
Redux Toolkit	`>=2.6.1`
React Router	`>=4.0.19`

Axios	`>=1.8.4`
Expo	`>52.0.41`

💻 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-webmobile2-sub1/S12P11A609.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.

```
EXPO_PUBLIC_API_URL=https://j12a601.p.ssafy.io/api
EXPO_PUBLIC_KAKAO_JS_KEY=903326f84d8fa9719394269d3021a521
EXPO_PUBLIC_KAKAO_REST_API_KEY=7e68504ad57f1e923c845f01bcf187
e9
```

4️⃣ 개발 서버 실행

로컬 개발 환경에서 프로젝트를 실행하려면 다음 명령어를 입력하세요.

```
# 프로젝트 빌드
eas build -p android --profile production
```

🔗 **실행 후:**

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

📦 프로젝트 빌드

배포용으로 프로젝트를 빌드하려면 다음 명령어를 실행하세요.

```
# 프로젝트 빌드
eas build -p android --profile production
```

BackEnd

- **Gradle : 8.11.1**
- **Redis : 7.4.2**
- **PostgreSQL : 7.2**
- **Spring Boot : 3.4.1**
- **Java : 17**
- **IntelliJ : 2024.3.1.1**

application.yml

```yaml
spring:
  config:
    import:
      #     - classpath:application-auth.yml
```

```
      - classpath:application-redis.yml
      - classpath:application-mail.yml
      - classpath:application-jwt.yml
      - classpath:application-mysql.yml
      - classpath:application-kakao.yml
      - classpath:application-s3.yml
      - classpath:application-cors.yml
      - classpath:application-api.yml
      - classpath:application-mongo.yml
      #     - classpath:application-auth.yml
      #     - classpath:application-firebase.yml
      - classpath:application-firebase.yml
      #     - classpath:application-mongo.yml
      #     - classpath:application-elastic.yml
      - classpath:application-ocr.yml
      - classpath:application-sms.yml
      - optional:file:.env[.properties]
  application:
    name: moba
  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

springdoc:
  swagger-ui:
    path: /swagger-ui.html
  api-docs:
```

```
    enabled: true
    path: /v3/api-docs
```

application-api.yml

```
Moba:
  bank:
    accesstoken: ${MOBA_BANK_ACCESSTOKEN}
    account: ${MOBA_BANK_ACCOUNT}
    base:
      url: ${MOBA_BANK_URL}
#  mydata:
#    base:
#      url: ${MOBA_MYDATA_URL}
```

application-cors.yml

```
cors:
  allowed-origins:
    - http://localhost:8081
    - https://j12a601.p.ssafy.io
    - http://localhost:8080
```

application-firebase.yml

```
spring:
  firebase:
    path: ${FCM_CREDENTIALS_PATH}
```

application-jwt.yml

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

application-kakao.yml

```
kakao:
  auth:
    client: ${KAKAO_CLIENT_ID}
    redirect: ${KAKAO_REDIRECT_URL}
    token-uri: https://kauth.kakao.com/oauth/token
    member-info-uri: https://kapi.kakao.com/v2/user/me
```

application-mail.yml

```
spring:
  mail:
    host: smtp.gmail.com
    port: 587
    username: ${MAIL_USERNAME}
    password: ${MAIL_PASSWORD}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
```

application-mongo.yml

```
spring:
  data:
    mongodb:
      host: ${MONGO_INITDB_HOST}
      port: ${MONGO_INITDB_PORT}
      database: ${MONGO_INITDB_DATABASE}
```

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-ocr.yml

```
clova:
  ocr:
    url: ${CLOVA_OCR_URL}
    secret: ${CLOVA_OCR_SECRET}

gpt:
  api:
    key: ${GPT_API_KEY}
```

application-redis.yml

```
spring:
  # Redis 설정
  data:
    redis:
      host: moba-redis
      port: 6379
      database: 0
```

application-s3.yml

```
cloud:
  aws:
```

```
  credentials:
    access-key: ${S3_ACCESS_KEY}
    secret-key: ${S3_SECRET_KEY}
  region:
    static: ap-northeast-2
  s3:
    bucket: ${S3_BUCKET_NAME}
```

application-sms.yml

```
coolsms:
  api:
    key: ${COOLSMS_API_KEY}
    secret: ${COOLSMS_API_SECRET}
```

bundle.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.3'
    id 'io.spring.dependency-management' version '1.1.7'
}

bootJar {
    archiveFileName = 'app.jar'
}

group = 'com.a601'
version = '0.0.1-SNAPSHOT'

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

configurations {
```

```
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    compileOnly 'org.projectlombok:lombok'
    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 'io.jsonwebtoken:jjwt-api:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-impl:0.11.5'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-mail'
    implementation 'io.github.cdimascio:dotenv-java:3.0.0'
    implementation 'org.springframework.boot:spring-boot-starter-json'
//    implementation 'org.springframework.boot:spring-boot-starter-tomcat'
//    implementation 'org.springframework:spring-web:6.1.12'
//    implementation 'org.springframework:spring-webmvc:6.1.14'

    //AWS S3
    implementation 'com.amazonaws:aws-java-sdk-s3:1.12.681'
    implementation 'javax.xml.bind:jaxb-api:2.3.1'

    //Swagger
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.3.0
```

```
    implementation 'org.hibernate.validator:hibernate-validator:8.0.1.Final'

    //firebase
    implementation 'com.google.firebase:firebase-admin:9.2.0'

    //websocket
    implementation 'org.springframework.boot:spring-boot-starter-websocket'

    //MongoDB
    implementation 'org.springframework.boot:spring-boot-starter-data-mongo

    //SMS
    implementation 'net.nurigo:sdk:4.3.0'

}


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