# 포팅 메뉴얼( 빌드 및 배포설정 )

| | |
|---|---|
| 태그 | 중요 문서 |
| 이벤트 시간 | |

기술 스택 & 버전 정보
빌드 방법
    Backend
    Frontend
배포 설정
기타 설정
    Backend

---

## 기술 스택 & 버전 정보

1. **이슈 관리 :**
    - **Jira**
2. **형상 관리:**
    - **Gitlab**
3. **커뮤니케이션:**
    - **Notion**
    - **MatterMost**
    - **Discord**
4. **개발 환경**
    - **IDE :**
        - **Intellij : 2022.3.1**
        - **Visual Studio Code : 1.74.2**
    - **DB :**
        - **MariaDB : 10.6.11**
    - **UI & UX :**
        - **Figma**
    - **Server :**
        - **AWS EC2 Ubuntu : 22.04**
        - **S3**
        - **Nginx : 1.23.3**
5. **상세**
    - **Backend :**
        - **JAVA : 11.0.17**
        - **Spring Boot : 2.7.9**
        - **Gradle : 7.6.1**
        - **FCM**

- **Frontend :**
  - **React : 18.2.0**
  - **web3 : 1.9.0**
  - **typescript : 2.1.4**
  - **firebase : 9.19.0**
- **메타버스 게임**
  - **Unity : 2021.3.9f**
- **블록체인 :**
  - **Geth : 1.11.6-unstable-b1acaf47**
  - **Ganache : 2.5.4**
  - **Truffle : 5.7.7**
- **메시지 처리:**
  - **RabbitMQ : 3.11.11**
  - **Redis : 7.0.10**
- **CI/CD :**
  - **Jenkins : 2.394**
  - **Docker : 23.0.1**
  - **DockerHub**

# 빌드 방법

## Backend

```
cd back
./gradlew build
```

## Frontend

```
npm install
npm run start
```

# 배포 설정

1. **docker 설치**

```
sudo apt-get update
sudo apt-get install \
   ca-certificates \
   curl \
   gnupg \
   lsb-release

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"
apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

1. **jenkins 설치용 DockerFile**

```
FROM jenkins/jenkins:latest # jenkins 이미지를 사용
USER root # root 유저로 진행

COPY docker_install.sh /docker_install.sh # docker_install.sh 파일을 복사
RUN chmod +x /docker_install.sh # 파일 권한에 실행이 가능하도록 추가
RUN /docker_install.sh # sh파일 실행
```

```
RUN usermod -aG docker jenkins # jenkins유저를 docker그룹에 추가
USER jenkins # jenkins 유저로 docker container 접근
```

1. **jenkins 내부에 docker 설치**

```
FROM jenkins/jenkins:latest # jenkins 이미지를 사용
USER root # root 유저로 진행

COPY docker_install.sh /docker_install.sh # docker_install.sh 파일을 복사
RUN chmod +x /docker_install.sh # 파일 권한에 실행이 가능하도록 추가
RUN /docker_install.sh # sh파일 실행

RUN usermod -aG docker jenkins # jenkins유저를 docker그룹에 추가
USER jenkins # jenkins 유저로 docker container 접근
```

1. **certbot 이미지 및 컨테이너 실행**

```
 sudo mkdir certbot
 cd certbot
 sudo mkdir conf www logs

 sudo docker pull certbot/certbot
 sudo docker run -it --rm --name certbot -p 80:80 \
             -v "/home/ubuntu/certbot/conf:/etc/letsencrypt" \
             -v "/home/ubuntu/certbot/log:/var/log/letsencrypt" \
             -v "/home/ubuntu/certbot/www:/var/www/certbot" \
             certbot/certbot certonly
```

1. **nginx 설정**

```
server {
        listen       80;
        listen  [::]:80;
        server_name  j8e207.p.ssafy.io;
        server_tokens off; # nginx 버전 노출 제한

        location / {
                return 301 https://$host$request_uri; # 80포트로 오는 요청 443포트로 redirect
        }

        location /.well-known/acme-challenge/ {
                root /var/www/certbot;
        }
}

server {
        include /etc/nginx/conf.d/service-url.inc;

        listen       443 ssl;
        listen  [::]:443 ssl;
        server_name  j8e207.p.ssafy.io;
        server_tokens off; # nginx 버전 노출 제한

        # SSL 설정
        ssl_certificate /etc/letsencrypt/live/j8e207.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j8e207.p.ssafy.io/privkey.pem;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;
        ssl_ciphers ALL;

        location / {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
                try_files $uri $uri/ /index.html; # url에 대한 파일 없으면 index.html 반환
        }

        location /api/v1 {
                resolver 172.26.0.2;
                proxy_pass $service_url;

                proxy_redirect     off;

                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-Host $server_name;
```

```
                proxy_set_header   X-Forwarded-Proto https;
        }

        location /rpc {
                proxy_redirect off;
                proxy_pass    http://j8e207.p.ssafy.io:8545/;
        }
}
```

1. **Blue Green 배포를 위한 BE 동작서버 관리파일 ( service-url.inc )**

```
set $service_url http://j8e207.p.ssafy.io:8082;
```

1. **FE DockerFile**

```
FROM node:latest as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build
```

2. **FE deploy-front.sh**

```
sudo docker rmi e207/front:1.0
sudo docker pull e207/front:1.0

docker run -d -p 3000:3000 --name front e207/front:1.0

docker cp front:/usr/src/app/build /home/ubuntu/dev/html

sudo cp -r /home/ubuntu/dev/html/build/* /home/ubuntu/dev/html
sudo rm -r /home/ubuntu/dev/html/build
```

3. **FE Jenkins**

```
cd front
docker build -t e207/front:1.0 .
echo $PASSWORD | docker login -u $USERNAME --password-stdin
docker push e207/front:1.0
docker rmi e207/front:1.0

docker image prune
```

```
echo $PASSWORD | docker login -u $USERNAME --password-stdin
cd dev
sh deploy-front.sh
```

1. **BE Dockerfile**

```
FROM openjdk:11-jdk
VOLUME /tmp
ARG JAR_FILE=./build/libs/back-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","app.jar", "--spring.profiles.active=${Profile}"]
```

2. **BE deploy-back.sh**

```
#!/bin/bash

echo "> 현재 구동중인 profile 확인"
CURRENT_PROFILE=$(curl -X POST https://j8e207.p.ssafy.io/api/v1/profile)

echo "> $CURRENT_PROFILE"

if [ $CURRENT_PROFILE == v1 ]
then
        IDLE_PROFILE=v2
        IDLE_PORT=8082
        IDLE_TAG=2.0
        CURRENT_TAG=1.0
elif [ $CURRENT_PROFILE == v2 ]
then
        IDLE_PROFILE=v1
        IDLE_PORT=8081
        IDLE_TAG=1.0
        CURRENT_TAG=2.0
else
        echo "> 일치하는 Profile이 없습니다. Profile: $CURRENT_PROFILE"
        IDLE_PROFILE=v1
        IDLE_PORT=8081
        IDLE_TAG=1.0
        CURRENT_TAG=2.0
fi



sudo docker ps -a -q --filter "name=back-${IDLE_PROFILE}" | grep -q . && docker stop back-$IDLE_PROFILE && docker rm back-$IDLE_PRO
FILE | true
sudo docker rmi e207/back:$IDLE_TAG
sudo docker pull e207/back:$IDLE_TAG
docker run -d -p $IDLE_PORT:${IDLE_PORT} --name back-$IDLE_PROFILE -e Profile=dev,$IDLE_PROFILE e207/back:$IDLE_TAG

# 정상구동 확인

sleep 10

for retry_count in {1..10}
do
  response=$(curl -X GET https://j8e207.p.ssafy.io/api/v1/actuator/health)
  up_count=$(echo $response | grep 'UP' | wc -l)

  if [ $up_count -ge 1 ]
  then # $up_count >= 1 ("UP" 문자열이 있는지 검증)
      echo "> Health check 성공"
      break
  else
      echo "> Health check의 응답을 알 수 없거나 혹은 status가 UP이 아닙니다."
      echo "> Health check: ${response}"
  fi

  if [ $retry_count -eq 10 ]
  then
    echo "> Health check 실패. "
    exit 1
  fi

  echo "> Health check 연결 실패. 재시도..."
  sleep 10
done


# 정상구동 성공, nginx 포트변경
echo "> 전환할 Port : $IDLE_PORT"
echo "> Port 전환"
echo "set \$service_url http://j8e207.p.ssafy.io:${IDLE_PORT};" | sudo tee /home/ubuntu/dev/conf.d/service-url.inc

echo "> Nginx Reload"
docker exec nginx service nginx reload
```

3. **BE Jenkins**

```
cd back
chmod +x gradlew
./gradlew clean build -Pprofile=dev test

CURRENT_PROFILE=$(curl -X POST https://j8e207.p.ssafy.io/api/v1/profile)
if [ $CURRENT_PROFILE == v1 ]
then
    IDLE_TAG=2.0
elif [ $CURRENT_PROFILE == v2 ]
```

```
then
    IDLE_TAG=1.0
else
        echo "> 일치하는 Profile이 없습니다. Profile: $CURRENT_PROFILE"
    IDLE_TAG=1.0
fi

docker build -t e207/back:$IDLE_TAG .
echo $PASSWORD | docker login -u $USERNAME --password-stdin

docker push e207/back:$IDLE_TAG
docker rmi e207/back:$IDLE_TAG

docker image prune
```

```
echo $PASSWORD | docker login -u $USERNAME --password-stdin
cd dev
sh deploy-back.sh
```

1. **PROD Jenkins**

```
cd back
chmod +x gradlew
./gradlew clean build -Pprofile=prod test

CURRENT_PROFILE=$(curl -X POST https://onthemars.site/api/v1/profile)
if [ $CURRENT_PROFILE == v1 ]
then
    IDLE_TAG=2.0
elif [ $CURRENT_PROFILE == v2 ]
then
    IDLE_TAG=1.0
else
        echo "> 일치하는 Profile이 없습니다. Profile: $CURRENT_PROFILE"
    IDLE_TAG=1.0
fi

docker build -t e207/back-prod:$IDLE_TAG .
docker build -t e207/front:latest ../front

echo $PASSWORD | docker login -u $USERNAME --password-stdin
docker push e207/back-prod:$IDLE_TAG
docker rmi e207/back-prod:$IDLE_TAG

docker push e207/front:latest
docker rmi e207/front:latest
```

```
echo $PASSWORD | docker login -u $USERNAME --password-stdin
sh deploy.sh
```

# 기타 설정

## Backend

1. **build.gradle**

```
plugins {
  id 'java'
  id 'org.springframework.boot' version '2.7.9'
  id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'onthemars'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
  compileOnly {
    extendsFrom annotationProcessor
  }
```

```
}

repositories {
  mavenCentral()
}

dependencies {
  developmentOnly 'org.springframework.boot:spring-boot-devtools'

  implementation 'org.springframework.boot:spring-boot-starter-web'
  implementation 'org.springframework.boot:spring-boot-starter-actuator'
  implementation 'org.springframework.boot:spring-boot-starter-validation'

  implementation 'com.google.firebase:firebase-admin:6.8.1'
  implementation group: 'com.squareup.okhttp3', name: 'okhttp', version: '4.2.2'

  // rabbit mq
  implementation 'org.springframework.boot:spring-boot-starter-amqp'
  implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310'

  // DB
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.springframework.boot:spring-boot-starter-data-redis'
  implementation 'org.mariadb.jdbc:mariadb-java-client:3.1.2'
  implementation 'com.mysql:mysql-connector-j:8.0.32'

  // security
  implementation 'org.springframework.boot:spring-boot-starter-security'
  testImplementation 'org.springframework.security:spring-security-test:5.7.5'

  // swagger
  implementation 'io.springfox:springfox-boot-starter:3.0.0'

  // web3j
  implementation 'org.web3j:core:5.0.0'

  //  jwt
  implementation group: 'io.jsonwebtoken', name: 'jjwt-api', version: '0.11.2'
  runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-impl', version: '0.11.2'
  runtimeOnly group: 'io.jsonwebtoken', name: 'jjwt-jackson', version: '0.11.2'

  // s3
  implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'

  // 파일 확장자 검증
  implementation 'org.apache.tika:tika-core:2.6.0'

  annotationProcessor 'org.projectlombok:lombok'
  compileOnly 'org.projectlombok:lombok'


  testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

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

def gradleProfile = project.hasProperty("profile") ? project.property("profile").toString() : "default"

test {
  systemProperty("spring.profiles.active", gradleProfile)
}

task copyPrivate(type: Copy){
  copy{
    from '../s08p22e207-sub'
    include '*.yml'
    include 'firebase_service_key.json'
    include '*.sql'
    into 'src/main/resources'
  }
}

task copyPrivateTest(type: Copy) {
  copy {
    from '../s08p22e207-sub'
    include '*.yml'
    into 'src/test/resources'
  }
}
```

2. **application.yml**

```
server:
  port: 8080
  servlet:
    context-path: /api/v1/
  forward-headers-strategy: framework

spring:
  rabbitmq:
    host: localhost
    port: 5672
    username: {유저이름}
    password: {비밀번호}

  redis:
    host: localhost
    port: 6379

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/onthemars?characterEncoding=UTF-8&serverTimezone=Asia/Seoul
    username: {유저이름}
    password: {비밀번호}

  sql:
    init:
      mode: always

  jpa:
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        # show_sql: true
    open-in-view: false
    defer-datasource-initialization: true

  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 100MB

logging:
  level:
    com:
      amazonaws:
        util:
          EC2MetadataUtils: error
    org:
      hibernate:
        type:
          descriptor:
            sql: TRACE
        SQL: DEBUG
cloud:
  aws:
    s3:
      bucket: onthemars-dev
    stack:
      auto: false
    region:
      static: ap-northeast-2
    credentials:
      accessKey: { s3 accessKey }
      secretKey: { s3 secretKey }

jwt:
  secret: jwt-E207-otmarsbuk-jwt-E207-otmarsbuk-jwt-E207-otmarsbuk-jwt-E207-otmarsbuk-jwt-E207-otmarsbuk-jwt-E207-otmarsbuk-jwt-E207-o
```

3. **application-dev.yml**

```
spring:
  redis:
    host: j8e207.p.ssafy.io
    port: 6379

  rabbitmq:
```

```
    host: j8e207.p.ssafy.io
    port: 5672
    username: {유저이름}
    password: {비밀번호}

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://maria-dev.c3dw6kz0sgd4.ap-northeast-2.rds.amazonaws.com:3306/onthemars?characterEncoding=UTF-8&serverTimezone=
    username: {유저이름}
    password: {비밀번호}

  sql:
    init:
      mode: never

  jpa:
    database-platform: org.hibernate.dialect.MariaDB106Dialect
    hibernate:
      ddl-auto: validate
---
server:
  port: 8081

spring:
  config:
    activate:
      on-profile: v1
---
server:
  port: 8082

spring:
  config:
    activate:
      on-profile: v2
```

4. **application-prod.yml**

```
spring:
  redis:
    host: onthemars.site
    port: 6379

  rabbitmq:
    host: onthemars.site
    port: 5672
    username: {유저이름}
    password: {비밀번호}

  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://maria-prod.c3dw6kz0sgd4.ap-northeast-2.rds.amazonaws.com:3306/onthemars?characterEncoding=UTF-8&serverTimezone
    username: {유저이름}
    password: {비밀번호}

  sql:
    init:
      mode: never

  jpa:
    database-platform: org.hibernate.dialect.MariaDB106Dialect
    hibernate:
      ddl-auto: validate

springdoc:
  api-docs:
    enabled: false
  swagger-ui:
    enabled: false

cloud:
  aws:
    s3:
      bucket: onthemars
    stack:
      auto: false
    region:
      static: ap-northeast-2
    credentials:
      accessKey: {s3 accessKey}
      secretKey: {s3 secretKey}
---
```

```
server:
  port: 8081

spring:
  config:
    activate:
      on-profile: v1
---
server:
  port: 8082

spring:
  config:
    activate:
      on-profile: v2
```