

# 나랏말싸피 포팅 매뉴얼

본 문서는 나랏말싸피를 사용하기 위한 가이드로 Front-End에 React, Back-End에 Spring Boot, FastAPI, Node Express를 배포하는 과정에 대해 서술하고 있습니다.

## 1. 개발 환경

### 1.1. React 주요 기술스택 / 버전정보

- React 18.2.0
- React-axios 2.0.6
- React-Bootstrap 5.2.1
- React redux 8.2.0
- React toolkit 1.8.5
- React dnd 16.0.1
- React howler 5.2.0
- React icons 4.4.0
- Redux-persist 6.0.0

### 1.2. Spring Boot 주요 기술스택 / 버전정보

- Spring Boot 2.7.5
- java 1.8
- Gradle 7.5.1
- Hibernate 5.6.12
- lombok 1.18.24
- swagger 3.0.0
- java-jwt 3.10.3

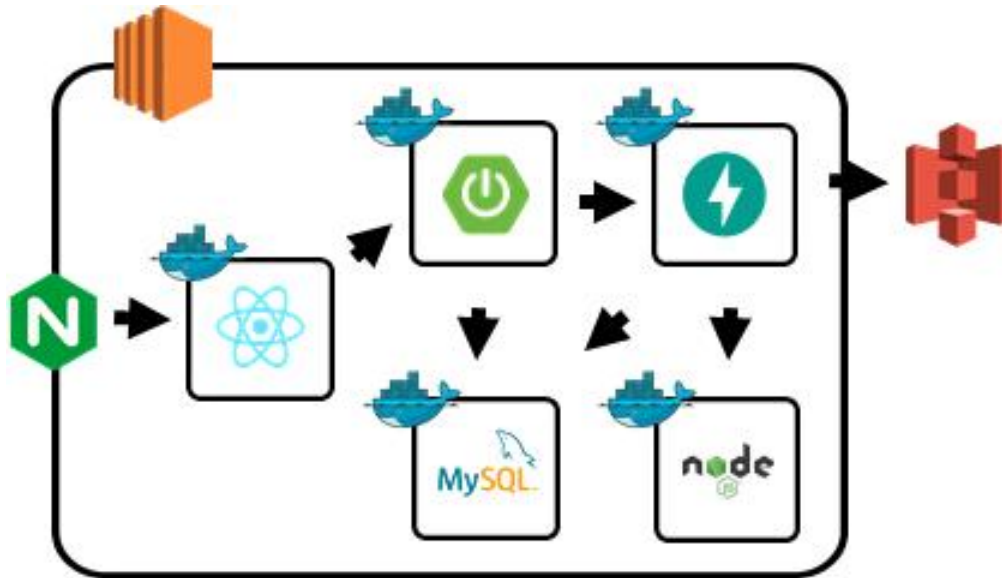
### 1.3. FastAPI 주요 기술스택 / 버전정보

- FastAPI 0.83.0
- Python 3.6.13
- torch 1.10.0
- numpy 1.19.5
- scikit-learn 0.24.2
- FastAPI-SQLAlchemy 0.1.0
- boto3 1.23.10
- opencv-python 4.6.0.66
- uvicorn 0.16.0

#### 1.4. Node Express 주요 기술스택 / 버전정보

- node 16.18.0(LTS)
- node express 4.18.2
- graceful-fs 4.2.10
- pngjs 3.4.0
- shelljs 0.8.5
- svg2ttf 6.0.3
- svgicons2svgfont 9.2.0
- (global) ttf2woff 3.0.0

## 2. 아키텍처 구성도



< 그림 1 : 아키텍처 구성도 >

## 3. 배포 설정

\* apt 혹은 apt-get install의 경우 update를 먼저 수행해야하는 경우가 종종 있는데 작성 편의상 생략한다.

\* 한글도메인 나랏말싸피 사용으로 유니코드가 적용되는 부분이 있다. 유니코드에서는 나랏말싸피.com을 xn--910b35kqzb51p93w.com으로 표현한다.

### 3.1. 버전 정보

- 서버 : Ubuntu 20.04.4 LTS
- Docker : 20.10.17
- Nginx : 1.18.0(ubuntu)
- Certbot : 0.40.0
- Jenkins : 2.367

### 3.2. 도메인 설정

#### 3.2.1. 가비아 발급 도메인 적용

나랏말싸피.com | 등록 확인중 | 인증 코드

등록일: 2022-11-03 | 만기일: 2023-11-03 (생분 기간: 349일) | 만기 전 삭제

만기일 맞춤 | 도메인 연장 | 연장 알림

소유자 | 수정 | 소유권 이전

관리자 | 수정

대입서버 | 설정

1차	ns.gabia.com
2차	ns1.gabia.com
3차	ns.gabia.net
4차	대입된 값
5차	대입된 값
6차	대입된 값
7차	대입된 값
8차	대입된 값
9차	대입된 값
10차	대입된 값
11차	대입된 값
12차	대입된 값
13차	대입된 값

보안 서비스

등록정보 숨김 | 설정

안전 잠금 | 설정

기관 이전 잠금 | 설정

가비아 보안 | 관리

백지서신 보안 | 설정

DNS 정보

도메인 연결 | 설정

포워딩 | 설정

확산 | 설정

DNS 레코드 설정 | 설정

< 그림 2 : 가비아 도메인 관리 페이지 >

3.2.1.1. 가비아에서 발급받은 도메인 관리 페이지 하단의 DNS 레코드 설정에 들어간다.

DNS 관리

나랏말싸피.com

레코드 개수: 1개 | 최근 업데이트: 2022-11-03 12:38:29 | 내입서버: ns.gabia.com

이력 확인 | 액셀 다운로드

DNS 설정 | 레코드 수정

타입	호스트	값/위치	TTL	우선 순위	서비스
CNAME	@	k7d110.p.ssafy.io.	3600		DNS 설정

3.2.1.2. DNS 관리에서 타입을 CNAME, 호스트는 @, 값/위치는 (발급받은 EC2 도메인) + '.'으로 설정한다. 이를 통해 나랏말싸피.com을 주소창에 입력하면 k7d110.p.ssafy.io으로 가게 된다.

### 3.3. Certbot ssl 인증서

#### 3.3.1. Certbot 설치

- apt-get install certbot 으로 설치한다.

#### 3.3.2. Certbot 인증서 발급

\* 루트권한이 필요하다.

\* nginx가 작동중이라면 standalone 방식의 발급을 받을 수 없다. 만약 작동중이라면 (sudo) service nginx stop 또는 (sudo) systemctl stop nginx로 nginx의 작동을 잠시 멈추고, 작업 이후 다시 start해준다.

- (sudo) certbot --standalone -d k7d110.p.ssafy.io xn--910b35kqzb51p93w.com certonly 명령어로 ec2의 기존 도메인과 나랏말싸피.com 둘 다 적용되도록 설정한다.

- 이메일주소나 Y 등을 입력하면 완료되며 private key와 fullchain key가 저장된다.

### 3.4. Nginx 설정

```
server {
    # SSL configuration
    #
    listen 443 ssl default_server; # for ipv4
    listen [::]:443 ssl default_server; # for ipv6

    client_max_body_size 100M;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    ssl_certificate /etc/letsencrypt/live/k7d110.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k7d110.p.ssafy.io/privkey.pem;

    server_name XN--910B35KQZB51P93W.COM;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass http://localhost:3000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }

    location /api {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass http://localhost:8081;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }

    location /fastapi {
        proxy_pass http://127.0.0.1:18080;
```

```

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;

        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        proxy_connect_timeout 3000s;
        proxy_read_timeout 6000s;
        proxy_send_timeout 6000s;
    }

    location /nodeexpress {
        proxy_pass http://localhost:28080;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
    }
}

server {
    listen 443;
    server_name k7d110.p.ssafy.io;
    root html;

    location / {
        return 301 https://XN--910B35KQZB51P93W.COM$request_uri;
    }
}

server {
    listen 80;
    server_name XN--910B35KQZB51P93W.COM k7d110.p.ssafy.io;
    root html;

    location / {
        return 301 https://XN--910B35KQZB51P93W.COM$request_uri;
    }
}

```

- (초록색 음영) 앞서 발급한 ssl 인증서를 등록한다.
- (노란색 음영) 나랏말싸피.com이라는 주소의 접근에 대해 관리한다.
- (연보라색 음영) 443부분 : https://k7d110.p.ssafy.io/의 접근을 https://나랏말싸피.com/으로 리다이렉트 한다.

- (연보라색 음영) 80부분 : <http://k7d110.p.ssafy.io/>, <http://나랏말싸피.com/>의 접근을 <https://나랏말싸피.com/>으로 리다이렉트한다.
- fastapi나 nodeexpress의 경우 내부 도커네트워크 호출을 하는 것이 안전하나, 원래 gpu서버를 받았다면 외부호출을 해야했기도 했고 개발과 테스트 편의를 위해 외부에서 직접호출이 가능하도록 열어두었다.

### 3.5. Mysql 배포 설정

#### 3.5.1. Mysql Docker image 다운받기

- (sudo) docker pull mysql:8.0.31

#### 3.5.2. Mysql docker-compose.yml 설정

```
version: '3' # docker compose 버전
services:
  local-db:
    image: library/mysql:8.0.31
    container_name: naratmalssafy_mysql # 컨테이너 이름
    restart: always
    ports:
      - 13306:3306 # 로컬의 13306 포트를 컨테이너의 3306포트로 연결
    environment:
      MYSQL_USER: k7d110mal
      MYSQL_PASSWORD: k7d110mysqlpw
      MYSQL_ROOT_PASSWORD: k7d110mysqlrootpw!@#
      TZ: Asia/Seoul
    volumes:
      - ./db/mysql/data:/var/lib/mysql
      - ./db/mysql/init:/docker-entrypoint-initdb.d
```

- 다운받은 8.0.31버전 이미지로 mysql 컨테이너를 생성한다.
- volume mount를 통해 저장된 DB가 (EC2가 날라가지 않는 한) container 재시작시에도 소멸되지 않게 한다.

### 3.6. Jenkins 설정

\* React, Spring Boot, FastAPI, Node Express는 지속적인 개발이 이루어지기 때문에 Jenkins를 통한 CI/CD를 구축한다.

\* 4가지 모두 DockerFile과 docker-compose.yml을 통해 실행할 예정이라 공통부분 위주로 작성한다.

### 3.6.1. Jenkins 가입

- k7d110.p.ssafy.io:9090으로 들어가서 jenkins의 admin계정을 만들어준다.

- 이때, 인증을 요구하는데 화면에 보이는 ec2 파일을 열어보면 인증코드가 있어서 복사/붙여넣기를 하면 된다.

### 3.6.2. plugin 설치

- 가입 전 설치되는 standard를 제외하고도 Dashboard에서 gitlab, gradle 관련 plugin을 모두 설치한다.

### 3.6.3. gitlab 연결설정

- 1) 새로운 아이템을 freestyle으로 선택한다.
- 2) 소스코드 관리는 다음과 같이 설정한다.



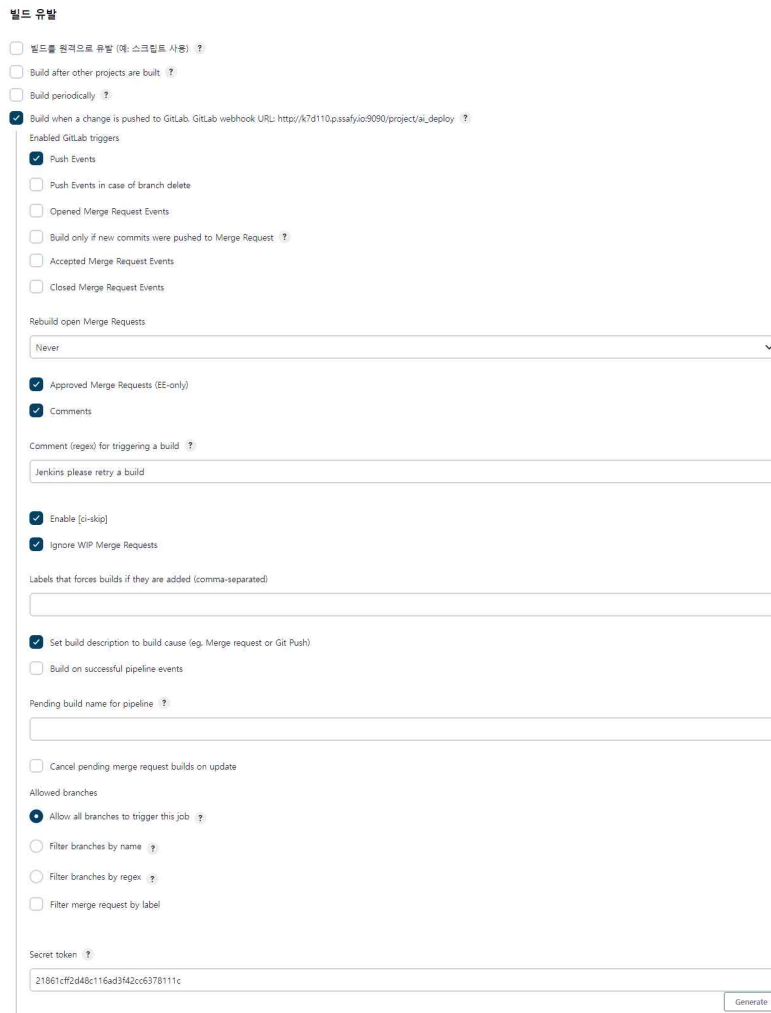
< 그림 4 : Jenkins 소스코드 관리 탭 >

- deploy\_AI 로 시작하는 모든 branch에 대해 감지하겠다는 것이다. 이를 생각하여 다른 branch의 접두사가 되는 branch name이 없도



록 네이밍을 해주면 좋다.

3) 빌드 유발은 다음과 같이 설정한다.



The image shows the 'Build Triggers' configuration page in Jenkins. The 'Build when a change is pushed to GitLab' option is selected. Under 'Enabled GitLab triggers', 'Push Events' is checked. The 'Rebuild open Merge Requests' dropdown is set to 'Never'. 'Approved Merge Requests (EE-only)' and 'Comments' are checked. The 'Comment (regex) for triggering a build' field contains 'Jenkins please retry a build'. 'Enable [ci-skip]' and 'Ignore WIP Merge Requests' are checked. The 'Labels that forces builds if they are added (comma-separated)' field is empty. 'Set build description to build cause (eg. Merge request or Git Push)' is checked. 'Build on successful pipeline events' is unchecked. The 'Pending build name for pipeline' field is empty. 'Cancel pending merge request builds on update' is unchecked. Under 'Allowed branches', 'Allow all branches to trigger this job' is selected. The 'Secret token' field contains a long alphanumeric string, and a 'Generate' button is visible.

< 그림 5 : Jenkins 빌드 유발 설정 >

- 푸시 이벤트 (merge request가 적용되었을 때도 발생한다)가 발생하면 작동한다.

- Secret token은 gitlab에서 webhook의 secret token으로 사용하는 것이다.

## Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

### URL

URL must be percent-encoded if it contains one or more special characters.

### Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

### Trigger

☒ Push events

Push to the repository.

< 그림 6 : gitlab webhook 설정 >

4) Build Steps에서 Execute shell을 선택한다. 이는 git pull 이후 ec2 상에서 이루어질 동작을 말한다.

```
cd /var/lib/jenkins/workspace/ai_deploy/AI
cp /home/ubuntu/dockerFiles/AI/Dockerfile /var/lib/jenkins/workspace/ai_deploy/AI/Dockerfile
cp /home/ubuntu/dockerComposes/AI/docker-compose.yml /var/lib/jenkins/workspace/ai_deploy/AI/docker-compose.yml
docker ps -f name=fastapi-compose -q | xargs --no-run-if-empty docker container stop
docker container ls -a -f name=fastapi-compose -q | xargs -r docker container rm
docker-compose up --force-recreate --build -d
docker image prune -f
```

- 내용은 위와 비슷한 형태로 작성한다.

- 원래 git pull 이후에는 해당 branch의 루트 (ai\_deploy)에 가는데, git 상에 AI폴더 안에 DockerFile과 docker-compose.yml파일이 있어 cd 명령어로 위치를 바꿔준다.

- cp 명령어로 DockerFile과 docker-compose.yml을 ec2 로컬에서 복사해오는 것은 코드상에 변화없이 ec2에서 안정화가 되면 추후 git에 반영하기 위해서이다. 주석처리할 경우 git에 있는 파일로 실행된다.

- fastapi-compose라는 이름을 가진 container를 멈추고 삭제한 다음 docker-compose.yml 파일을 통해 다시 만든다. 이후 기존 이미지는 삭제한다.

- Spring Boot의 경우에는 앞에 조금 추가되는 부분이 있어 추가 작성한다.

```
chmod 755 /var/lib/jenkins/workspace/backend_deploy/main/BE/gradlew
cd /var/lib/jenkins/workspace/backend_deploy/main/BE/
/var/lib/jenkins/workspace/backend_deploy/main/BE/gradlew clean build -Pprofile=deploy -x test
```

- git pull 시 gradlew가 executable하지 않기 때문에 executable한

상태로 바꿔주고, build할 때 application.properties에서 배포용 deploy가 선택되도록 한다.

### 3.7. docker-compose.yml 설정

\* docker-compose는 여러 컨테이너를 사용할 경우 유용하다. 하지만 해당 기능보다는 관리의 편안함을 위해 사용했다.

\* 추가로 docker-compose.yml 파일 자체는 4가지 다 대동소이하여 React를 대표하여 작성한다.

```
version: "3"
services:
  react:
    user: root
    container_name: react-compose
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 3000:3000
    volumes:
      - /var/lib/jenkins/FONT:/FONT
    extra_hosts:
      - "host.docker.internal:host-gateway"
    networks:
      - inner-network
  networks:
    inner-network:
      external: true
```

- FONT라는 폴더의 내용을 지속시키고 다른 컨테이너와 쉽게 공유하기 위하여 volume mount한 특성이 있다.

- 다른 docker의 경우 외부 라이브러리의 코드 수정 등을 위해 특정 파일 단위 mount들도 있다. (fastapi의 torch cpu설정용, nodeexpress의 fontsvg table 정보 수정용 등)

### 3.8. DockerFile 설정

- DockerFile은 도커 이미지를 만들기 위한 파일이고 도커 이미지가 만들어질 때 명령어를 순차적으로 수행한다.

#### 1) React

```
FROM node:16.18.0
WORKDIR /usr/src/app
```

```
COPY package.json ./
COPY ./ ./
RUN npm install --save --legacy-peer-deps
RUN npm run build
RUN yarn global add serve
COPY ./ ./
CMD [ "npm", "run", "start" ]
```

## 2) Spring Boot

```
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

## 3) FastAPI

```
From python:3.6.13

COPY ./ ./AI/

WORKDIR ./AI/app

EXPOSE 18080

RUN apt-get update
RUN apt-get -y install libgl1-mesa-glx

RUN pip install --use-deprecated=legacy-resolver -r requirements.txt

RUN pip install --use-deprecated=legacy-resolver -r fontmaker/requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "18080"]
```

- 주의사항 : fastapi의 경우 --host로 외부 접근을 허용하지 않으면 container 내부에서의 localhost접근만 허용된다.

## 4) Node Express

```
From node:16.18.0

COPY ./ ./node_server/

WORKDIR ./node_server
```

```
RUN npm install
```

```
CMD ["node", "app.js"]
```

## 4. 기타 설정

### 4.1. S3 저장소 설정

- ttf, woff 파일 저장을 위한 s3 서버가 필요하다.

- 1) AWS 리전 ap-northeast-2로 생성한다. (버킷명 : naraymalssafy)
- 2) 버킷 정책을 다음과 같이 설정한다.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1666676706808",
  "Statement": [
    {
      "Sid": "Stmt1666676700880",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::naratmalssafy/*"
    },
    {
      "Sid": "2",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity E1DWNFEBUGK0JX"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::naratmalssafy/*"
    }
  ]
}
```

- 3) CORS 세팅은 다음과 같이 한다.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
```

```

        "GET",
        "HEAD"
    ],
    "AllowedOrigins": [
        "https://xn--910b35kqzb51p93w.com",
        "https://k7d110.p.ssafy.io"
    ],
    "ExposeHeaders": [
        "x-amz-server-side-encryption",
        "x-amz-request-id",
        "x-amz-id-2"
    ],
    "MaxAgeSeconds": 3000
}
]

```

## 4.2. CloudFront 설정

- S3의 경우 Origin Header를 적절하지 못하게 받는 경우가 있다.  
이를 해결하기 위해 CloudFront를 사용한다.

1) 배포를 생성한다.

- 원본 도메인 : `naratmalssafy.s3.ap-northeast-2.amazonaws.com`
- 사용자 정의 헤더 추가 : Origin \*

2) 생성 후 설정에서 사용자 정의 SSL 인증서를 선택한다.

- 당연히 처음엔 없을 것이고, 인증서 요청을 눌러 생성해야한다.
- 완전히 정규화된 도메인 이름에 유니코드 `xn--910b35kqzb51p93w.com`을 입력한다.
- 검증 방법은 이메일 검증을 하여 빠르게 검증한다. (DNS 검증은 수십분에서 수 시간 이상 걸린다고 한다.)