

포팅 매뉴얼

목차

1. 프로젝트 기술 스택
2. Frontend
3. Backend
 - A. Spring
 - B. Django
4. AWS EC2
 - A. SSH 설정
 - B. Docker
 - C. Jenkins
5. Jenkins
6. Docker
 - A. Frontend
 - B. Django
 - C. Backend
7. Nginx
8. 외부 서비스 정보
 - A. 카카오 공유
 - B. Youtube API
 - C. STEAM API
 - D. AWS S3

1. 프로젝트 기술 스택

형상관리	GitLab		.
이슈관리	Jira		.
UX/UI	Figma		.
프로젝트 일정 관리	Notion		.
빌드 및 배포 관리	AWS EC2	Jenkins	2.387.1
		Docker	23.0.1
		Docker compose	1.25.0
		Nginx	1.18.0
		Certbot	.
IDE	IntelliJ		2021.2.4
	VSCode		1.75.1
Database	MySQL		8.0.32
	MongoDB		6.0.5
	Redis		7.0.10
프론트엔드	Node.js		18.12.1
	yarn		1.22.19
	VSCode IDE EXTENSION	ESLint	2.4.0
		Prettier	0.10.4
	React		17.0.2
	React-router-dom		6.8.2
	recoil		0.7.7
백엔드	Spring	Springboot	2.7.7
		Lombok	.
		Spring security	.
		JPA	.
		Gradle	7.6
		Jdk	Zulu-11
		Swagger	3.0.0
	Django	Django	3.2.13
		python	3.11
		pandas	1.5.3

2. Frontend

패키지 설치 및 실행

yarn install

yarn start

빌드 및 배포

Dockerfile을 통해 진행

3. Backend

Spring

환경변수 설정

`Edit Configurations` 상단 + 를 눌러 다음의 환경변수를 추가

AWS_ACCESS_KEY

AWS_SECRET_KEY

JWT_SECRET_KEY

STEAM_API_KEY

빌드 및 배포

Dockerfile을 통해 진행

Django

가상환경 설정 및 실행

python -m venv venv

source venv/Script/activate

pip install -r requirements.txt

python manage.py makemigrations

python manage.py migration

python manage.py runserver

빌드 및 배포

Dockerfile을 통해 진행

4. AWS EC2

SSH

방화벽 설정

```
sudo ufw allow ssh  
sudo ufw enable
```

Docker

Apt가 HTTPS를 통해 repository를 이용하는 것을 허용할 수 있도록 해주는 패키지들 설치

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

Docker 공식 GPG key 추가 및 repository 등록 및 설치

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

Jenkins

Docker에 Jenkins 설치 및 구동

```
docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v  
/var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:its
```

5. Jenkins

플러그인 설치

다음의 플러그인 설치

GitLab

Publish Over SSH

자동 빌드 및 배포 설정

Jenkins React execute shell

```
cd front-end  
docker-compose up -d --build --force-recreate  
docker image prune -f
```

Jenkins Springboot execute shell

```
cd backend/gamemakase
docker-compose up -d --build --force-recreate
docker image prune -f
```

Jenkins Django execute shell

```
cd django
docker build -t django .
docker ps -q --filter "name=django" && docker stop django && docker rm django
docker run -d -p 5000:8000 -e TZ=Asia/Seoul -it --cpus 4 --memory 30g --name django django
```

환경변수 등록

Jenkins 관리 -> 시스템 설정 -> Global properties -> Environment variables

AWS_ACCESS_KEY / AWS_SECRET_KEY / JWT_SECRET_KEY

STEAM_API_KEY / REACT_APP_KAKAO_JAVASCRIPT_KEY 등록

6. Docker

Frontend

React dockerfile

```
FROM node:18.12.1-alpine as builder
WORKDIR "/app"
COPY package.json yarn.lock ./
RUN yarn install --production=false
COPY . .
ENV PATH /front-end/node_modules/.bin:$PATH
RUN yarn build
FROM nginx
RUN mkdir /app
WORKDIR /app
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
COPY --from=builder /app/build /app/build
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

Nginx.conf

```
server {  
    listen 3000;  
    location / {  
        root    /app/build;  
        index   index.html;  
        try_files $uri $uri/ /index.html;}}
```

Dockercompose.yml

```
version: "3.7"  
services:  
    frontend:  
        container_name: frontend  
        build:  
            context: .  
            dockerfile: ./Dockerfile  
        ports:  
            - "3000:3000"  
        restart: always  
        environment:  
            SERVER_MODE: prod  
            REACT_APP_KAKAO_JAVASCRIPT_KEY : ${REACT_APP_KAKAO_JAVASCRIPT_KEY}  
            TZ: ASIA/SEOUL
```

Django

Django dockerfile

```
FROM python:3  
WORKDIR .  
COPY requirements.txt ./  
RUN apt-get update && apt-get install -y supervisor  
RUN pip install -r requirements.txt  
COPY . .  
EXPOSE 8000  
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000", "--noreload"]
```

Backend

Springboot dockerfile

```
FROM node:18.12.1-alpine as builder
WORKDIR "/app"
COPY package.json yarn.lock ./
RUN yarn install --production=false
COPY . .
ENV PATH /front-end/node_modules/.bin:$PATH
RUN yarn build
FROM nginx
RUN mkdir /app
WORKDIR /app
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
COPY --from=builder /app/build /app/build
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

Dockercompose.yml

```
version: "3.7"
services:
  redis:
    image: redis
    container_name: redis
    hostname: redis
    ports:
      - "6379:6379"
  springboot:
    container_name: springboot
    build:
      context: .
      dockerfile: ./Dockerfile
    ports:
      - "8080:8080"
    environment:
      SERVER_MODE: prod
      AWS_ACCESS_KEY: "${AWS_ACCESS_KEY}"
      AWS_SECRET_KEY: "${AWS_SECRET_KEY}"
      JWT_SECRET_KEY: "${JWT_SECRET_KEY}"
      STEAM_API_KEY: "${STEAM_API_KEY}"
```

7. Docker

/etc/nginx/nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
}
http {
    upstream server-application {
        server gamemakase.com:8080;
    }
    upstream client-application {
        server gamemakase.com:3000;
    }
    upstream recommend-application {
        server gamemakase.com:5000;
    }

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    server {
        listen 80;
        server_name gamemakase.com;

        location / {
            proxy_buffering off;
            return 301 https://gamemakase.com$request_uri;
        }
    }

    server {
        server_name gamemakase.com;

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/gamemakase.com-0001/fullchain.pem; #
managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/gamemakase.com-0001/privkey.pem;
# managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

        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 X-Forwarded-Proto https;
        proxy_headers_hash_bucket_size 512;
        proxy_redirect off;

        # Websockets
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
```



```

# Server application requests
location /api/ {
    proxy_pass http://server-application;
}

# Server application requests
location /pub/ {
    proxy_pass http://server-application;
}

location /auth/ {
    proxy_pass http://server-application;
}

# Server application requests
location /sub/ {
    proxy_pass http://server-application;
}

# Server application requests
location /ws {
    proxy_pass http://server-application;
}

# Server application requests
location ~ ^/(swagger-ui|webjars|configuration|swagger-resources|v2|csrf) {
    proxy_pass http://server-application;
}

# Client application requests
location /{
    proxy_pass http://client-application;
}

# Django application requests
location /recommend/ {
    proxy_pass http://recommend-application;
}

}

```

외부 서비스 정보

카카오 인증

카카오 개발자 사이트인 <https://developers.kakao.com/> 사이트를 이용합니다

사용할 애플리케이션을 식별이 가능하도록 서비스명으로 등록합니다. 애플리케이션을 만들면 앱키가 발급됩니다. 그 중에서 JavaScript 키를 복사합니다.

애플리케이션에서 Web 플랫폼을 사용하도록 등록합니다. 이 때 우리 서비스의 도메인인 <https://gamemakase.com> 을 등록합니다. (필요하다면 localhost 서버도 등록합니다.)

.env 파일에 복사해둔 JavaScript 키를 작성해둔 후, React 에서 process.env.를 통해 앱키를 불러와 Kakao.init 함수를 실행 합니다.

유튜브 API

유튜브 개발자 사이트를 이용하여 API 키를 발급받습니다.

(<https://console.cloud.google.com/apis/library/youtube.googleapis.com?hl=ko&project=still-resource-367503>)

사용자 인증 정보 메뉴에서 우측 상단의 API KEY 를 통해 키를 복사하여 사용 할 수 있습니다.

STEAM API

다음의 공식 문서를 참고하여 진행합니다.

(https://developer.valvesoftware.com/wiki/Steam_Web_API#GetPlayerSummaries_.28v0001.29)

아래 링크를 통해 API 키를 발급받습니다. (<https://steamcommunity.com/dev/apikey>)
스팀 계정이 있어야 하며, 해당 계정에 게임이 있어야 발급이 가능합니다.

AWS S3

서비스 내에서 사용되는 이미지 등의 자료를 공유하기 위해 AWS S3 클라우드 스토리지를 적용했습니다.