포팅 매뉴얼

목차

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:lts

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}"
```