포팅 매뉴얼

목차

1. 프로젝트 기술 스택

형상관리	GitLab		.
이슈관리	Jira		.
UX/UI	Figma		.
프로젝트 일정 관리	Notion		.
빌드 및 배포 관리	AWS EC2		2.387.2
	Docker		23.0.4
	Docekr-compsoe		1.25.0
	Nginx		1.18.0
	Certbot		.
IDE	intelliJ		2021.2.4
	Vscode		1.75.1
Database	MySQL		8.0.31
	MongoDB		6.0.5
	Redis		7.0.10
프론트엔드	React-Native		0.71.6
	Redux		4.2.1
	Styled-Components		5.3.9
	TypeScript		4.8.4
	Node.js		18.15.0
	Npm		9.5.0
백엔드	Spring	springboot	2.7.10
		Lombok	.
		Spring security	.
		JPA	.
		gradle	7.6
		jdk	Zulu-11
		swagger	3.0.0
	django	django	3.2.13
		python	3.11
		tensorflow	2.12.0
		apscheduler	0.6.2
		Googletrans	4.0.0
		PyJWT	2.6.0

2. Frontend

PC 실행 시

환경변수 설정

/frontend, /frontend/android 폴더에 .env 파일 추가

파일 내부에 KAKAO_APP_KEY 추가

패키지 설치 및 실행

npm install

npm start

모바일 실행 시

.apk 파일을 이용해서 앱 설치

3. Backend

Spring

환경변수 설정

`Edit Configurations` 상단 + 를 눌러 다음의 환경변수를 추가

OAUTH2_KAKAO_ID / OAUTH2_KAKAO_SECRET

빌드 및 배포

Dockerfile 을 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

Dockerfile 을 통해 진행

Billing

빌드 및 배포

수동 배포 진행

4. AWS EC2

SSH

방화벽 설정

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

Docker

Apt 가 HTTPS 를 통해 repository 를 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

Docker 에 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 Springboot execute shell

```
cd backend-business
cd FEELINGFILLING
docker-compose up -d –build
docker images prune –a
```

Jenkins Django execute shell

```
cd backend-ai
docker-compose up -d –build
docker images prune –a
```

환경변수 등록

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

DJANGO_SECRET_KEY / OAUTH2_KAKAO_ID / JWT_SECRET_KEY

OAUTH2_KAKAO_SECRET / OPEN_AI_API_KEY 등록

6. Docker

Django

Django dockerfile

```
FROM python:3
WORKDIR .

COPY requirements.txt ./
RUN apt-get update && apt-get install -y supervisor
RUN pip install --upgrade pip
RUN pip install --upgrade setuptools
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000", "--noreload"]
```

Django docker-compose.yml

```yaml
version: "3.7"

services:
  django:
    build:
      context: .
      dockerfile: ./Dockerfile
    container_name: django
    environment:
      SERVER_MODE: prod
      DJANGO_SECRET_KEY: ${DJANGO_SECRET_KEY}
      JWT_SECRET_KEY: ${JWT_SECRET_KEY}
      OPEN_AI_API_KEY: ${OPEN_AI_API_KEY}
    ports:
      - "8000:8000"
```

Backend

Springboot dockerfile

```dockerfile
FROM adoptopenjdk/openjdk11:jdk-11.0.10_9-alpine AS builder
WORKDIR /app
COPY . .
RUN chmod +x ./gradlew
RUN ./gradlew clean bootJar
FROM adoptopenjdk/openjdk11:jdk-11.0.10_9-alpine
COPY --from=builder /app/build/libs/*.jar app.jar

EXPOSE 8080
ENTRYPOINT ["java", "-Duser.timezone=Asia/Seoul", "-jar","/app.jar"]
```

Dockercompose.yml

```yaml
version: "3.7"

services:
  redis:
    image: redis
    container_name: redis
    hostname: redis
    ports:
      - "6379:6379"
    command: redis-server --requirepass mammoth77 --port 6379
  springboot:
    container_name: springboot
    build:
      context: .
      dockerfile: ./Dockerfile
    ports:
      - "8080:8080"
    environment:
      - TZ=Asia/Seoul
```

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;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        upstream server-application {
                server 127.0.0.1:8080;
        }

        upstream feelings-application {
                server 127.0.0.1:8000;
        }

        server{
                if ($host = feelingfilling.store) {
                        return 301 https://$host$request_uri;
                } # managed by Certbot


                listen 80;
                listen [::]:80;
                server_name feelingfilling.store;
                return 404; # managed by Certbot
        }

        server {
        server_name feelingfilling.store;

        listen 443 ssl;
        ssl_certificate /etc/letsencrypt/live/feelingfilling.store/fullchain.pem; #
managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/feelingfilling.store/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;


        # 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;
        }

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


        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

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

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript
text/xml application/xml application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

8. 외부 서비스 정보

Open Ai API

Open Ai 홈페이지에 가입을 합니다.
https://platform.openai.com/account/billing/overview

API 키를 사용하기 위해 billing 결제 정보를 등록합니다.

좌측 User > API keys 에서 API 키를 발급받습니다.

https://platform.openai.com/docs/models 공식문서를 통해 코드를 작성합니다.

Open API Vito

홈페이지로 이동하여 가입을 합니다. https://developers.vito.ai

API 키를 발급받은 다음, 서비스를 사용합니다.

Kakao login

Kakao developers 에 가입합니다. https://developers.kakao.com/

내 애플리케이션으로 이동하여 애플리케이션을 추가합니다.

앱 아이콘, 앱 이름, 사업자명을 입력합니다.

앱 설정 > 플랫폼에서 패키지명, 디버그 키 해시, 릴리즈 해시 키를 등록합니다.

앱 키를 조회하여 필요한 키를 사용합니다.