👀

포팅 매뉴얼

시연 시나리오

1. application.properties

```
# MariaDB configuration
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://54.180.141.143:3306/cook
spring.datasource.username=root
spring.datasource.password=ssafy

# jpa관련 설정
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# spring security관련 설정
spring.security.user.name=ssafy
spring.security.user.password=ssafy

# Server setup
#server.address=127.0.0.1
server.port=8080

#https를 위한 path설정
server.servlet.context-path=/api

# OAuth
spring.profiles.include=oauth
```

application-oauth.properties

```
## KAKAO Login
spring.security.oauth2.client.registration.kakao.client-id=0d4240c77e9221fb9d99142127676902
#spring.security.oauth2.client.registration.kakao.client-secret=secret키
spring.security.oauth2.client.registration.kakao.redirect-uri=https://i5b206.p.ssafy.io/oauth/callback/kakao
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.scope=profile_nickname, profile_image
spring.security.oauth2.client.registration.kakao.client-name=kakao
```

```
spring.security.oauth2.client.registration.kakao.client-authentication-method=POST
## kAKAO Provider
# 인가 코드 받기
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize
# 토큰 받기 (POST)
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
# 사용자 정보 가져오기
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=id
```

application.yml

```
springdoc:
  api-docs:
    groups:
      enabled: true
  swagger-ui:
    path: /swagger-ui.html
    displayRequestDuration: true
    groups-order: DESC
```

프론트 엔드

현재 배포 과정은 밑의 Jenkins 파일의 과정과 같이 배포되고 있다.

```
pipeline {
  agent none
  tools {nodejs 'nodejs'}
  stages {
    stage('Build') {
      agent {
        docker {
          image 'node:16-alpine'
        }
      }
      steps {
        dir ('frontend') {
          sh 'rm -f package-lock.json'
          sh 'yarn install'
          sh 'yarn build'
        }
      }
    }
    stage('Docker build') {
      agent any
      steps {
        dir ('frontend') {
          sh 'docker build -t licipe:front .'
        }
      }
    }
    stage ('Docker run') {
      agent any
      steps {
        dir ('frontend') {
          sh 'docker ps -f name=licipe -q | xargs --no-run-if-empty docker container stop'
          sh 'docker container ls -a -fname=licipe -q | xargs -r docker container rm'
          sh 'docker run -d --name licipe -p 8000:80 licipe:front'
        }
```

```
      }
    }
  }
  post {
    success {
      echo 'done'
    }
  }
}
```

Nodejs는 16 버전에서 작동시켰다.

git clone을 받았으면 frontend 폴더에서 오류 방지를 위해 package-lock.json 폴더를 삭제한 후,

의존성 설치를 위한 yarn install을 한 뒤 build를 한다.

```
cd frontend
rm -f package-lock.json
yarn install
yarn build
```

빌드한 파일을 사용하여 docker 이미지를 만든다.

```
docker build -t licipe:front .
```

docker 이미지의 중복을 피하기 위해 중복되는 이름의 이미지를 정지하고 삭제한 뒤, 다시 이미지를 작동시킨다.

```
docker ps -f name=licipe -q | xargs --no-run-if-empty docker container stop
docker container ls -a -fname=licipe -q | xargs -r docker container rm
docker run -d --name licipe -p 8000:80 licipe:front
```

현재 i5b206.p.ssafy.io:8000 uri에 프론트 엔드가 적용된 상황이다. 다음으로 도커의 정적 파일들을 nginx의 root 경로에 위치시켜준 뒤, nginx를 다시 불러온다.

```
rm -rf ~/dist/html
docker cp licipe:/usr/share/nginx/html ~/dist
sudo nginx -s reload
```

nginx 의 설정값은 /etc/nginx/conf.d/nginx.conf 파일에 기록되어 있다.

```
server {
        listen 443 ssl default_server;
```

```
        server_name i5b206.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/i5b206.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i5b206.p.ssafy.io/privkey.pem;

        location / {
                root /home/ubuntu/dist/html;
                proxy_pass http://i5b206.p.ssafy.io:8000;
                try_files $uri $uri/ /index.html$is_args$args;
        }
        location /api {
                proxy_pass http://i5b206.p.ssafy.io:8080;
        }
}

server {
        listen 80;
        rewrite ^(.*) https://i5b206.p.ssafy.io$request_uri;
}
```

특이사항

프론트엔드 도커 이미지 작동 후 i5b206.p.ssafy.io 사이트에서 Unexpected token error '<'가 발생할 수 있다. 그런 경우 잠시 기다린 후 정적 파일들을 불러오는 명령어들을 실행 하면 해결 된다.

```
rm -rf ~/dist/html
docker cp licipe:/usr/share/nginx/html ~/dist
sudo nginx -s reload
```

위의 방법으로 해결되지 않을 경우 nginx의 listen80 부분 서버 블록을 삭제하면 된다.

```
server {
        listen 443 ssl default_server;
        server_name i5b206.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/i5b206.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i5b206.p.ssafy.io/privkey.pem;

        location / {
                root /home/ubuntu/dist/html;
                proxy_pass http://i5b206.p.ssafy.io:8000;
                try_files $uri $uri/ /index.html$is_args$args;
        }
        location /api {
                proxy_pass http://i5b206.p.ssafy.io:8080;
        }
}
```

.env 파일을 사용하였다. git에 포함되어있다.

```
REACT_APP_API_URL = https://i5b206.p.ssafy.io
REACT_APP_API_PORT = /api

REACT_APP_KAKAO_REST_API_KEY = 0d4240c77e9221fb9d99142127676902
REACT_APP_KAKAO_REDIRECT_URI = https://i5b206.p.ssafy.io/oauth/callback/kakao

REACT_APP_AWS_IDENTITY_POOL_ID = ap-northeast-2:0e8878ad-e2d9-4d26-ba4d-d45f545a16d4
```

```
REACT_APP_AWS_S3_BUCKET = b206
REACT_APP_AWS_REGION = ap-northeast-2
```

백엔드

자바 버전

```
intelliJ ultimate 2021.2.1

java 11.0.11 2021-04-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.11+9-LTS-194)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.11+9-LTS-194, mixed mode)
```

AWS EC2 서버에 스프링부트 배포

1. EC2 서버에 프로젝트 Clone 받기

```
git 설치
sudo yum install git

git 버전 확인
git --version

프로젝트 디렉토리 생성
mkdir ~/app && mkdir ~/app/step1

프로젝트 디렉트리 이동
cd ~/app/step1

git clone
git clone 주소

프로젝트 확인
cd 프로젝트명
ll

단위 테스트만 수행
./gradlew test

실행 권한 추가
chmod +x ./gradlew

단위 테스트만 수행
./gradlew test
```

2. 배포 스크립트 만들기

- git pull을 통해 새 버전의 프로젝트를 받음

- Gradle을 통해 프로젝트 테스트와 빌드
- EC2 서버에서 해당 프로젝트 실행 및 재실행

```
deploy.sh 생성
vim /home/ubuntu/cook/S05P13B206/deploy.sh

deploy.sh 작성
#!/bin/bash

REPOSITORY=/home/ubuntu/cook
PROJECT_NAME=S05P13B206/backend

cd $REPOSITORY/$PROJECT_NAME/

echo "> Git Pull"

git pull origin develop

echo "> 프로젝트 Build 시작"
pwd
./gradlew build

echo "> step1 디렉토리 이동"

cd $REPOSITORY/S05P13B206/backend/build/libs

echo "> 현재 구동중인 애플리케이션 pid 확인"
pwd
ls
CURRENT_PID=$(pgrep -f cook-0.0.1-SNAPSHOT.jar)

echo "현재 구동 중인 애플리케이션 pid: $CURRENT_PID"

if [ -z "$CURRENT_PID" ]; then
   echo "> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다."
else
   echo "> kill -15 $CURRENT_PID"
   kill -15 $CURRENT_PID
   sleep 5
fi

echo "> 새 애플리케이션 배포"

nohup java -jar $REPOSITORY/$PROJECT_NAME/build/libs/cook-0.0.1-SNAPSHOT.jar &


deploy.sh 실행 권한 추가
chmod +x ./deploy.sh

deploy.sh 실행
./deploy.s
```

외부 서비스 정보

1. 카카오 로그인 설정
    - KAKAO DEVELOPER 가입 및 애플리케이션 생성

    앱설정

    1. 일반 → 사업자 등록 X

2. 앱키 → `./frontend/.env`
3. 플랫폼 → Web: 사이트 도메인 설정

```
http://localhost:3000
https://localhost:3000
http://i5b206.p.ssafy.io
https://i5b206.p.ssafy.io
http://i5b206.p.ssafy.io:8000
https://i5b206.p.ssafy.io:8000
```

제품설정/카카오 로그인

1. 활성화 설정: 상태 ON

   REDIRECT URI / 백엔드와 같은 주소 사용해야함

```
http://localhost:3000/oauth/callback/kakao
http://i5b206.p.ssafy.io/oauth/callback/kakao
http://i5b206.p.ssafy.io:8000/oauth/callback/kakao
http://i5b206.p.ssafy.io/api/oauth/callback/kakao
https://localhost:3000/oauth/callback/kakao
https://i5b206.p.ssafy.io:8000/oauth/callback/kakao
https://i5b206.p.ssafy.io/api/oauth/callback/kakao
https://i5b206.p.ssafy.io/oauth/callback/kakao
https://i5b206.p.ssafy.io:8000/api/oauth/callback/kakao
```

2. 동의항목

   닉네임 **필수**

   프로필사진 **필수**

   카카오 계정(이메일) **선택**: 비즈니스 앱인 경우에만 필수 가능

2. AWS S3: SNS 기능
   a. `https://s3.console.aws.amazon.com/s3/home` 에서 버킷 생성
   b. 권한: SNS 기능이므로 모두 비활성

```
모든 퍼블릭 액세스 차단: 비활성
새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단: 비활성
임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단:  비활성
새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단:  비활성
임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단: 비활성
```

   c. 버킷 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::b206/*"
```

```
        }
    ]
}
```

d. 피부여자

```
버킷 소유자(AWS 계정)
객체: 나열, 쓰기
버킷 ACL: 읽기, 쓰기
```

e. CORS(Cross-origin 리소스 공유)

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE",
            "GET",
            "HEAD"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

DB 생성 : AWS + Docker + MariaDB

AWS EC2 root 계정

PW: ssafy

MariaDB 계정

ID: root

PW: ssafy

1. AWS에 Docker설치(ubuntu 기준)

`su-` : root 권한 획득

- 처음 이라면 root계정 비밀번호 초기화 필요`sudo passwd root`

이동후, 도커 설치

`apt install docker.io`

`service docker start`

`docker info` or `docker version` 으로 설치 확인

2. Docker에 MariaDB설치

`docker search mariadb` : 마리아디비 이미지 서치

`docker pull mariadb/server:latest` : 이미지 pull

`:latest` 자리에 원하는 버전코드 입력해서 다운가능

`docker images` 로 설치된 이미지 파일 확인

3. Docker 컨테이너 생성

`docker run --name mariadb -e MARIADB_ROOT_PASSWORD=ssafy -p 3306:3306 -d mariadb/server:latest`

> 하면 id 값 리턴 해줌

`docker ps -a` 생성된 컨테이너 확인

```
docker restart mariadb

docker stop mariadb

docker start mariadb
# stop의 경우에는 마리아db가 먼저 종료
docker kill mariadb
```

4. MariaDB 컨테이너 진입

`docker exec -i -t mariadb bash` : 도커 마리아디비 컨테이너 접속

`mysql -uroot -pssafy` : 아이디 비밀번호 입력

`show databases;` 로 확인

`use mysql;`

`select host, user, password from user;`

> localhost뒤에 %의 뜻은 어디서든 접속이 가능하다라는뜻

`CREATE DATABASE [DB이름];` : 디비 생성 (우리는 'cook')

도커안에서 `/etc/mysql/my.cnf` 파일을 vim으로 열어서 `bind-address = 127.0.0.1`앞에 주석이없다면 주석처리해주고 저장하고 나온다.

`/etc/mysql/mariadb.conf.d/50-server.cnf` 에서 설정을 해줘야 하는 경우도 있다.

port설정도 필요하다면 이곳에서 하면된다. `port = 1234`

도커 재시작`service docker restart mariadb`

public ip address: 54.180.141.143

> curl ifconfig.me

DBeaver등을 통해 연결 확인

```
// application.properties

spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://{public-ip}:3306/cook
spring.datasource.username=root
spring.datasource.password=비밀번호
```