카공조아 - 포팅 메뉴얼

I. 빌드 및 배포

1. 개발 환경

구분	이름	version
Front-end	React JavaScript Node (npm) Nvm	18.2 ES6 (with Babel) 18.13.0 1.1.10
Back-end	SpringBoot Java JPA Lombok gson	2.7.7 JDK 11 1.18.24 2.10.1
Database	MySQL Redis	8.0.30 7.0.8
AUTH	JWT Kakao OAUTH	com.auth0 (4.2.2)
API	Kakao Map	
Infra (CI/CD)	Git lab AWS Zenkins Docker nginx	aws os: ubuntu 20.04 jenkins: 2.375.2 docker: 23.0.0 nginx: 1.18.0

2. 설정 파일 목록

- 스프링부트
  - build.gradle
  - application.yml
  - application-local.yml
  - application-prod.yml
  - 원격 환경변수 (/home/ubuntu/myEnv/springEnv)
- 리액트
  - package.json
  - .env
  - .env.production

3. 설정 파일 및 환경 변수 정보

1. 스프링 부트

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '2.7.8'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'
```

```
configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    // https://mvnrepository.com/artifact/com.google.code.gson/gson
    implementation group: 'com.google.code.gson', name: 'gson', version: '2.10.1'

    // https://mvnrepository.com/artifact/com.auth0/java-jwt
    implementation group: 'com.auth0', name: 'java-jwt', version: '4.2.2'

    // https://mvnrepository.com/artifact/org.hibernate/hibernate-spatial
    implementation group: 'org.hibernate', name: 'hibernate-spatial', version: '6.1.6.Final', ext: 'pom'

    // https://mvnrepository.com/artifact/org.locationtech.jts/jts-core
    implementation group: 'org.locationtech.jts', name: 'jts-core', version: '1.19.0'

    // https://mvnrepository.com/artifact/io.awspring.cloud/spring-cloud-starter-aws
    implementation group: 'io.awspring.cloud', name: 'spring-cloud-starter-aws', version: '2.4.3'

    // https://mvnrepository.com/artifact/org.qlrm/qlrm
    implementation group: 'org.qlrm', name: 'qlrm', version: '3.0.4'


}

tasks.named('test') {
    useJUnitPlatform()
}
```

application.yml

```
spring:
  profiles:
    active: prod
```

application-local.yml

```
server:
  port: 9000
  servlet:
    context-path: /api

spring:
  redis:
    lettuce:
      pool:
        max-active: 10
        max-idle: 10
        min-idle: 2

    port: 6379
    host: 127.0.0.1
    password: 'aaaa'

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
```

```
    url: jdbc:mysql://localhost:3306/ssafya308cafe?serverTimezone=Asia/Seoul&tinyInt1isBit=false
    username: ssafy
    password: ssafy

  jpa:
#    database: mysql
#    database-platform: org.hibernate.spatial.dialect.mysql.MySQL56InnoDBSpatialDialect
    open-in-view: true
    hibernate:
      ddl-auto: none
      naming:
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
      use-new-id-generator-mappings: false
    show-sql: true
    properties:
      hibernate.format_sql: false
#     dialect: org.hibernate.dialect.MySQL5InnoDBDialect

logging:
  level:
    org.hibernate.SQL: debug

kakaoOauth:
  REST_API_KEY: 204f458585e0229e8443cd7bc1be5c5e
  REDIRECT_URL: http://localhost:3000/oauth/kakao

jwt:
  # base64로 인코딩된 암호 키, HS512를 사용할 것이기 때문에, 512비트(64바이트) 이상이 되어야 합니다. 길게 써주세요
  secretKey: c3NhZnk46riwMu2Vmeq4sOqzte2Gte2UhOuhnOygne2KuEEzMDg=

  access:
    expiration: 6000

  refresh:
    expiration: 6000

cloud:
  aws:
    credentials:
      access-key: AKIA5F6XJ4YINB47D5DH
      secret-key: 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
    s3: #버킷이름
      bucket: cafestudyjoa
    region: #S3 지역
      static: ap-northeast-2
    stack:
      auto: false
```

application-prod.yml

```
server:
  port: ${A308_SERVER_PORT}
  servlet:
    context-path: /api

spring:
  redis:
    lettuce:
      pool:
        max-active: 10
        max-idle: 10
        min-idle: 2

    port: 6379
    host: ${A308_REDIS_HOST}
    password: ${A308_REDIS_PW}

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${A308_MYSQL_URL}
    username: ${A308_MYSQL_USER}
    password: ${A308_MYSQL_PW}

  jpa:
    #    database: mysql
    #    database-platform: org.hibernate.spatial.dialect.mysql.MySQL56InnoDBSpatialDialect
    open-in-view: true
    hibernate:
```

```
      ddl-auto: none
      naming:
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
      use-new-id-generator-mappings: false
    show-sql: true
    properties:
      hibernate.format_sql: true
#      dialect: org.hibernate.dialect.MySQL5InnoDBDialect

logging:
  level:
    org.hibernate.SQL: debug

kakaoOauth:
  REST_API_KEY: ${A308_REST_API_KEY}
  REDIRECT_URL: ${A308_REDIRECT_URL}

jwt:
  # base64로 인코딩된 암호 키, HS512를 사용할 것이기 때문에, 512비트(64바이트) 이상이 되어야 합니다. 길게 써주세요
  secretKey: ${A308_JWT_KEY}

  access:
    expiration: 100 # 분

  refresh:
    expiration: 20000 # 분

cloud:
  aws:
    credentials:
      access-key: ${A308_ACCESS_KEY}
      secret-key: ${A308_SECRET_KEY}
    s3: #버킷이름
      bucket: cafestudyjoa
    region: #S3 지역
      static: ap-northeast-2
    stack:
      auto: false
```

Spring의 application.yml에 들어가는 환경 변수 목록

```
A308_SERVER_PORT=9000
A308_REDIS_PORT=6379
A308_REDIS_HOST=redis
A308_REDIS_PW=9485ssafy308
A308_MYSQL_URL=jdbc:mysql://mysql:3306/ssafya308cafe?allowPublicKeyRetrieval=true&useSSL=false&tinyInt1isBit=false
A308_MYSQL_USER=ssafy
A308_MYSQL_PW=9485ssafy308
A308_REST_API_KEY=204f458585e0229e8443cd7bc1be5c5e
A308_REDIRECT_URL=https://i8a308.p.ssafy.io/oauth/kakao
A308_JWT_KEY=c3NhZnk46riwMu2Vmeq4sOqzte2Gte2UhOuhnOygne2KuEEzMDg=
A308_ACCESS_KEY=AKIA5F6XJ4YINB47D5DH
A308_SECRET_KEY=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
```

React의 env파일에 들어가는 환경 변수 목록

```
# .env
REACT_APP_KAKAO_REST_API_KEY = '204f458585e0229e8443cd7bc1be5c5e'
REACT_APP_KAKAO_REDIRECT_URI ='http://localhost:3000/oauth/kakao'
REACT_APP_REST_DEFAULT_URL = 'http://localhost:9000/api'

# .env.production
REACT_APP_KAKAO_REST_API_KEY = '204f458585e0229e8443cd7bc1be5c5e'
REACT_APP_KAKAO_REDIRECT_URI ='https://i8a308.p.ssafy.io/oauth/kakao'
REACT_APP_REST_DEFAULT_URL = 'https://i8a308.p.ssafy.io/api'
```

[리액트 설정파일]

package.json

```
{
  "name": "saffy_vue",
  "version": "0.1.0",
  "private": true,
```

```
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint",
    "test": "jest",
    "test:unit": "jest --no-cache"
  },
  "dependencies": {
    "email-validator": "^2.0.4",
    "local-lib": "^0.1.0",
    "password-validator": "^5.0.3",
    "vue": "^2.6.10",
    "vue-router": "^3.1.3",
    "vuex": "^3.1.2"
  },
  "devDependencies": {
    "@babel/core": "^7.16.7",
    "@babel/preset-env": "^7.16.8",
    "@vue/cli-plugin-babel": "^4.5.15",
    "@vue/cli-plugin-eslint": "^5.0.0",
    "@vue/cli-service": "^4.5.15",
    "@vue/test-utils": "^1.3.0",
    "babel-eslint": "^10.0.3",
    "babel-jest": "^27.4.6",
    "babel-preset-env": "^1.7.0",
    "babel-preset-jest": "^25.0.0",
    "core-js": "^3.20.2",
    "eslint": "^7.5.0",
    "eslint-plugin-import": "^2.25.4",
    "eslint-plugin-node": "^11.1.0",
    "eslint-plugin-promise": "^6.0.0",
    "eslint-plugin-standard": "^5.0.0",
    "eslint-plugin-vue": "^8.2.0",
    "jest": "^24.9.0",
    "jest-transform-stub": "^2.0.0",
    "node-sass": "^6.0.0",
    "sass-loader": "^10.0.0",
    "vue-jest": "^3.0.7",
    "vue-template-compiler": "^2.6.10",
    "vue-test-utils": "^1.0.0-beta.11",
    "webpack": "^4.45.0"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true,
      "jest": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "rules": {
      "no-console": "off",
      "no-unused-vars": "off"
    },
    "parserOptions": {
      "parser": "babel-eslint"
    }
  },
  "browserslist": [
    "> 1%",
    "last 2 versions"
  ]
}
```

.env

```
REACT_APP_KAKAO_REST_API_KEY = '204f458585e0229e8443cd7bc1be5c5e'
REACT_APP_KAKAO_REDIRECT_URI ='http://localhost:3000/oauth/kakao'
REACT_APP_REST_DEFAULT_URL = 'http://localhost:9000/api'
```

.env.production

```
REACT_APP_KAKAO_REST_API_KEY = '204f458585e0229e8443cd7bc1be5c5e'
REACT_APP_KAKAO_REDIRECT_URI ='https://i8a308.p.ssafy.io/oauth/kakao'
REACT_APP_REST_DEFAULT_URL = 'https://i8a308.p.ssafy.io/api'
```

4. DB Infra 세팅 + Table 세팅

- mysql 설치

```
# 기존 설치된 mysql off 후 진행

docker run --name mysql -d -p 3306:3306 --network ssafy-network -e MYSQL_ROOT_PASSWORD=9485ssafy308 mysql:8

# 로컬일때
docker run --name mysql -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=ssafy mysql:8

# 워크벤치와 연결 후 사용자 추가 및 admin 권한 주기
```

- Redis 설치

```
# 원격에서 레디스 컨테이너 띄우기
docker run --name redis --network ssafy-network -p 6379:6379 -d redis --requirepass 9485ssafy308

# 로컬일때
docker run --name redis -p 6379:6379 -d redis

# cli로 접속하기
docker exec -it {컨테이너id} redis-cli
```

- mysql 설정

```


# mysql bash로 접속

docker exec -i -t mysql bash

mysql -u root -p


# 사용자 추가

use mysql;

select host, user from user;

create user 'ssafy'@'%' identified by '9485ssafy308';

select host, user from user;

flush privileges;

# 권한 부여

show grants for 'ssafy'@'%';

GRANT ALL PRIVILEGES ON *.* TO 'ssafy'@'%';

show grants for 'ssafy'@'%';
```

- mysql DDL

[1. 먼저 테이블 부터 생성한다]

```
create database ssafya308cafe;

use ssafya308cafe;

DROP TABLE IF EXISTS `member`;

CREATE TABLE `member` (
  `id`  BIGINT  NOT NULL primary key auto_increment,
  `nickname`  VARCHAR(20) NOT NULL,
  `created_at` timestamp not null default current_timestamp,
    `updated_at` timestamp not null default current_timestamp on update current_timestamp,
  `current_badge` BIGINT  NULL,
  `oauth_type`  VARCHAR(20) NOT NULL,
  `oauth_id`  BIGINT  NOT NULL
);


DROP TABLE IF EXISTS `cafe`;
```

```
CREATE TABLE `cafe` (
    `id` BIGINT NOT NULL primary key auto_increment,
    `cafe_code`    VARCHAR(50)    NOT NULL    COMMENT '고유값',
    `name`    VARCHAR(50)    NOT NULL    COMMENT 'ex) 스타벅스 강남R점, 개인카페이름',
    `brand_type`    VARCHAR(50)    NOT NULL    COMMENT 'ex) 스타벅스 -> 로고 띄우기용'
);

DROP TABLE IF EXISTS `cafe_visit_log`;

CREATE TABLE `cafe_visit_log` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `visited_at`    INT    NOT NULL    COMMENT '230207',
    `acc_time`    INT    NOT NULL    DEFAULT 0    COMMENT '1분마다 업데이트, 최대 120',
    `member_id`    BIGINT    NOT NULL,
    `cafe_id`    BIGINT    NOT NULL,
    `fortune_id`    BIGINT    NULL    COMMENT '다시뽑기 하면 갱신됨',
    `is_survey`    TINYINT(1)    NOT NULL    DEFAULT 0,
    `is_crowd_survey`    TINYINT(1)    NOT NULL    DEFAULT 0
);


DROP TABLE IF EXISTS `cafe_location`;

CREATE TABLE `cafe_location` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `cafe_id`    BIGINT    NOT NULL,
    `lat` decimal(18,10) not null,
    `lng` decimal(18,10) not null,
    `address`    VARCHAR(100)    NOT NULL
);


DROP TABLE IF EXISTS `post`;

CREATE TABLE `post` (
  `id`  BIGINT  NOT NULL primary key auto_increment,
  `member_id` BIGINT  NOT NULL,
  `content` VARCHAR(255)  NOT NULL,
  `type`  VARCHAR(50) NOT NULL  COMMENT '카테고리. 기본 자유글',
  `is_cafe_authorized` TINYINT(1)    NOT NULL    DEFAULT 0,
  `created_at` timestamp not null default current_timestamp,
    `updated_at` timestamp not null default current_timestamp on update current_timestamp
);

DROP TABLE IF EXISTS `post_img`;

CREATE TABLE `post_img` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `post_id`    BIGINT    NOT NULL,
    `img_url`    VARCHAR(255)    NOT NULL,
    `access_key`    VARCHAR(100)    NOT NULL
);


DROP TABLE IF EXISTS `comment`;

CREATE TABLE `comment` (
  `id`  BIGINT  NOT NULL primary key auto_increment,
  `post_id` BIGINT  NOT NULL,
  `member_id` BIGINT  NOT NULL,
  `content` VARCHAR(255)  NOT NULL,
  `created_at` timestamp not null default current_timestamp,
  `updated_at` timestamp not null default current_timestamp on update current_timestamp,
  `group_no`  BIGINT  NOT NULL  COMMENT '부모댓글의 pk',
  `step_no` BIGINT  NOT NULL  COMMENT '댓글 순서'
);


DROP TABLE IF EXISTS `post_likes`;

CREATE TABLE `post_likes` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `post_id`    BIGINT    NOT NULL,
    `member_id`    BIGINT    NOT NULL
);


DROP TABLE IF EXISTS `comment_likes`;

CREATE TABLE `comment_likes` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `comment_id`    BIGINT    NOT NULL,
    `member_id`    BIGINT    NOT NULL
);

DROP TABLE IF EXISTS `fortune`;
```

```
CREATE TABLE `fortune` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `content`    VARCHAR(255)    NOT NULL
);


DROP TABLE IF EXISTS `todo`;

CREATE TABLE `todo` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `cafe_visit_log_id`    BIGINT    NOT NULL,
    `content`    VARCHAR(100)    NOT NULL,
    `is_complete`    TINYINT(1)    NOT NULL    DEFAULT 0
);

DROP TABLE IF EXISTS `member_cafe_tier`;

CREATE TABLE `member_cafe_tier` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `member_id`    BIGINT    NOT NULL,
    `cafe_id`    BIGINT    NOT NULL,
    `exp`    BIGINT    NOT NULL    DEFAULT 0
);

DROP TABLE IF EXISTS `member_coin`;

CREATE TABLE `member_coin` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `coffee_bean_cnt`    INT    NOT NULL    DEFAULT 0,
    `coffee_cnt`    INT    NOT NULL    DEFAULT 0,
    `member_id`    BIGINT    NOT NULL
);


DROP TABLE IF EXISTS `survey`;

CREATE TABLE `survey` (
  `id`  BIGINT  NOT NULL primary key auto_increment,
  `member_id` BIGINT  NOT NULL,
  `cafe_id` BIGINT  NOT NULL,
  `reply_wifi`  VARCHAR(1)  NOT NULL  COMMENT 'G/N/B (Good/Normal/Bad)',
  `reply_power` VARCHAR(1)  NOT NULL  COMMENT 'G/N/B',
  `reply_toilet`  VARCHAR(1)  NOT NULL  COMMENT 'G/N/B',
  `reply_time`  TINYINT(1)  NOT NULL  COMMENT 'Y/N',
  `created_at` timestamp not null default current_timestamp
);


DROP TABLE IF EXISTS `cafe_crowd`;

CREATE TABLE `cafe_crowd` (
  `id`  BIGINT  NOT NULL primary key auto_increment,
  `cafe_id` BIGINT  NOT NULL,
  `crowd_value` INT NOT NULL,
  `created_at` timestamp not null default current_timestamp
);


DROP TABLE IF EXISTS `post_cafe`;

CREATE TABLE `post_cafe` (
    `id`    BIGINT    NOT NULL primary key auto_increment,
    `post_id`    BIGINT    NOT NULL,
    `cafe_location_id`    BIGINT    NOT NULL
);
```

[2. fk 설정 전에 데이터를 입력한다]

- cafe.sql 실행 → cafe 데이터 입력됨
- cafe_location.sql → cafe location 데이터 입력됨
- 다음 코드를 입력하여 point type의 컬럼 추가

```
ALTER TABLE `cafe_location` ADD COLUMN point POINT;
set sql_safe_updates=0;
UPDATE `cafe_location` SET point = POINT(lat, lng);

-- 아래 두개 카운트가 동일한지 확인!
select count(*) from cafe_location;
select count(*) from cafe;
```

[3. 더미데이터 넣기]

```
use ssafya308cafe;

-- fortune dummy data

insert into fortune(content) values ('알고보면 나도 참 매력적'),('오늘은 술이 땡기네'),('관계를 돈독히 하세요'),
('부족할 것 없는 날입니다'),('꼬인 것이 풀립니다'),('오늘도 내가 너무 고생이 많아'),
('사랑을 심는 하루네요.'),('아름다운 하루 만드세요.'),('대화가 필요한 날입니다.'),('하던 일에 결실이 보이네요!'),('자신의 의견을 관철하세요.'),('약속한 것은
('귀인 덕분에 위기를 넘깁니다.'),('결정과 선택은 서두르지 마세요.'),('고민은 버려야 합니다.'),('집중해야 할 시기입니다.'),('신경 쓰지 말고 마이웨이!'),('좋은
,('과감하게 시도하세요.'),('뭐든지 묵묵히 해야 합니다.'),('좋은 인연이 나타납니다.'),('나이는 숫자에 불과합니다.'),('연애운이 풍부합니다. 도전!'),('드디어 능
('직장에서 좋은 소식을 듣습니다.'),('좋은 소식이 찾아옵니다.'),('다이어트는 내일부터..'),('유쾌, 통쾌, 상쾌한 하루'),('모든 일에 신중해야 합니다.'),
('오늘따라 반가워'),('단비가 내리네'),('경쟁과 배려 중 선택하세요'),('끝까지 용기를 잃지 마세요.'),('저는 노는 것에 재능이 있습니다..');
```

[4. fk 설정]

```
ALTER TABLE `cafe_location` ADD CONSTRAINT `FK_cafe_TO_cafe_location_1` FOREIGN KEY (
    `cafe_id`
)
REFERENCES `cafe` (
    `id`
)
on delete cascade
on update cascade;


ALTER TABLE `post` ADD CONSTRAINT `FK_member_TO_post_1` FOREIGN KEY (
  `member_id`
)
REFERENCES `member` (
  `id`
)
on delete cascade
on update cascade;


ALTER TABLE `comment` ADD CONSTRAINT `FK_post_TO_comment_1` FOREIGN KEY (
  `post_id`
)
REFERENCES `post` (
  `id`
)
on delete cascade
on update cascade;

ALTER TABLE `comment` ADD CONSTRAINT `FK_member_TO_comment_1` FOREIGN KEY (
  `member_id`
)
REFERENCES `member` (
  `id`
)
on delete cascade
on update cascade;

ALTER TABLE `member_coin` ADD CONSTRAINT `FK_member_TO_member_coin_1` FOREIGN KEY (
    `member_id`
)
REFERENCES `member` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `member_cafe_tier` ADD CONSTRAINT `FK_cafe_TO_member_cafe_tier_1` FOREIGN KEY (
    `cafe_id`
)
REFERENCES `cafe` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `member_cafe_tier` ADD CONSTRAINT `FK_member_TO_member_cafe_tier_1` FOREIGN KEY (
    `member_id`
)
REFERENCES `member` (
    `id`
)on delete cascade
on update cascade;
```

```
ALTER TABLE `comment_likes` ADD CONSTRAINT `FK_member_TO_comment_likes_1` FOREIGN KEY (
    `member_id`
)
REFERENCES `member` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `comment_likes` ADD CONSTRAINT `FK_comment_TO_comment_likes_1` FOREIGN KEY (
    `comment_id`
)
REFERENCES `comment` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `post_likes` ADD CONSTRAINT `FK_member_TO_post_likes_1` FOREIGN KEY (
    `member_id`
)
REFERENCES `member` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `post_likes` ADD CONSTRAINT `FK_post_TO_post_likes_1` FOREIGN KEY (
    `post_id`
)
REFERENCES `post` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `post_img` ADD CONSTRAINT `FK_post_TO_post_img_1` FOREIGN KEY (
    `post_id`
)
REFERENCES `post` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `cafe_visit_log` ADD CONSTRAINT `FK_cafe_TO_cafe_visit_log_1` FOREIGN KEY (
    `cafe_id`
)
REFERENCES `cafe` (
    `id`
)on delete cascade
on update cascade;

ALTER TABLE `cafe_visit_log` ADD CONSTRAINT `FK_member_TO_cafe_visit_log_1` FOREIGN KEY (
    `member_id`
)
REFERENCES `member` (
    `id`
)
on delete cascade
on update cascade;


ALTER TABLE `survey` ADD CONSTRAINT `FK_member_TO_survey_1` FOREIGN KEY (
  `member_id`
)
REFERENCES `member` (
  `id`
)
on delete cascade
on update cascade;


ALTER TABLE `survey` ADD CONSTRAINT `FK_cafe_TO_survey_1` FOREIGN KEY (
  `cafe_id`
)
REFERENCES `cafe` (
  `id`
)
on delete cascade
on update cascade;


ALTER TABLE `cafe_crowd` ADD CONSTRAINT `FK_cafe_TO_cafe_crowd_1` FOREIGN KEY (
  `cafe_id`
)
REFERENCES `cafe` (
  `id`
)
on delete cascade
on update cascade;
```

```
ALTER TABLE `post_cafe` ADD CONSTRAINT `FK_post_TO_post_cafe_1` FOREIGN KEY (
    `post_id`
)
REFERENCES `post` (
    `id`
)on delete cascade
on update cascade;


ALTER TABLE `post_cafe` ADD CONSTRAINT `FK_cafe_location_TO_post_cafe_1` FOREIGN KEY (
    `cafe_location_id`
)
REFERENCES `cafe_location` (
    `id`
)on delete cascade
on update cascade;
```

5. Https

- SSL 인증서 발급 (80, 443 사용중일시 해당 서비스 종료하고 진행하여야 함)

```
sudo letsencrypt certonly --standalone -d i8a308.p.ssafy.io
```

6. 기타 Infra 세팅

Docker

- 최신 버전으로 설치

Nginx

- 설치

```
sudo apt update
sudo apt upgrade
sudo apt autoremove

sudo apt install nginx

sudo service start nginx
sudo service status nginx
```

- nginx 설정파일 생성
  - sudo nano /etc/nginx/sites-available/cagong.conf
- nginx 설정파일 작성

```
server {
    listen 80;
    listen [::]:80;
    server_name i8a308.p.ssafy.io;
    server_tokens off;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl;
    server_name i8a308.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/i8a308.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i8a308.p.ssafy.io/privkey.pem;

    location / {
        proxy_pass http://i8a308.p.ssafy.io:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api {
```

```
        proxy_pass http://i8a308.p.ssafy.io:9000;
        proxy_http_version 1.1;
        proxy_set_header Connection "upgrade";
        proxy_set_header Upgrade $http_upgrade;
        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-NginX-Proxy true;
    }
}
```

- sudo ln -s /etc/nginx/sites-available/cagong.conf /etc/nginx/sites-enabled/
- sudo nginx -t
- 설정파일 세팅 완료 후 sudo service nginx restart

방화벽 설정

```
sudo ufw status verbose
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
sudo ufw allow 9000
sudo ufw allow 3000

sudo ufw allow 8080
sudo ufw enable
sudo ufw status
netstat -nlpt
```

7. CI/CD

- 젠킨스 설치

```
1. java 설치되어 있는지 확인(java -version) 후 없으면 open-jdk 설치

sudo apt-get install openjdk-11-jdk

2. repository 키 추가

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

3. sources.list 에 추가

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

4. jenkins를 apt-get 으로 설치

sudo apt-get update && apt-get install jenkins

5. Jenkins 시작

sudo systemctl start jenkins

6. 혹시 방화벽 있으면 열기

sudo ufw allow 8080

7. http://IP:8080 으로 Jenkins 최초 접속. Administrator password는 아래에서 얻는다.

sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- 젠킨스 파이프라인 작성

```
pipeline { // 파이프라인의 시작
    // 스테이지 별로 다른 거
    agent any // 아무 젠킨스 agent나 써라

    triggers {
        pollSCM('*/3 * * * *') // 3분주기로 파이프라인 구동하겠다
    }
```

```
environment { // 파이프라인 안에서 쓸 환경변수들. 시스템 환경변수로 들어감
  AWS_DEFAULT_REGION = 'ap-northeast-2' // 서울
  HOME = '.' // Avoid npm root owned
}

// 본격적인 파이프라인 내용
stages { // 각 stage는 큰 단계
    stage('Prepare') { // 1. prepare stage, 깃 레포지토리를 다운로드 받음
        agent any // agent는 아무나

        steps {
            echo 'Clonning Repository'

            // git pull로 땡겨온다
            git url: 'https://lab.ssafy.com/s08-webmobile2-sub2/S08P12A308.git', // 내 git url
                branch: 'master', // 푸쉬할 브랜치, master로 하면 에러남. main으로 해야함
                credentialsId: 'A308Jenkins-DT' // git 크레덴셜 등록한 id
        }

        post {
            // If Maven was able to run the tests, even if some of the test
            // failed, record the test results and archive the jar file.
            success { // 성공시 실행
                echo 'Successfully Pulled Repository'
            }

            always { // 성공하던 실패하던 실행
              echo "i tried..."
            }

            cleanup { // post 내용이 모두 끝났을때 실행
              echo "after all other post condition"
            }
        }
    }

    stage('Stop and Remove Old Container - Front') {
        // 안되면 try catch 적용해보기
        steps {
            script{
                try {
                    sh 'docker stop $(docker ps -q --filter ancestor=frontend:latest)'
                    sh 'docker rm $(docker ps -a -q --filter ancestor=frontend:latest)'
                } catch (Exception e) {
                    echo "An error occurred: ${e}"
                }
            }
        }

        post {
        success {
              echo 'Stop and Remove success!'
        }
      }
    }

    stage('Bulid Frontend') {
      // 도커 빌드
      agent any
      steps {
        echo 'Build Frontend'

        dir ('./frontend'){
            sh """
            docker build . -t frontend:latest
            """
        }
      }
    post {
        // steps 끝나면 post온다
        // 빌드하다 실패하면 error 뱉고, 나머지 과정 하지말고 종료
        failure {
          error 'This pipeline stops here...'
        }
      }
    }


    stage('Deploy New Frontend Container') {
        steps {
            sh 'docker run -p 3000:80 -d frontend:latest'
        }

        post {
        success {
              echo 'Deploy Frontend success!'
```

```
            }
          }
        }

        stage('Stop and Remove Old Container - Back') {
            // 안되면 try catch 적용해보기
            steps {
                script{
                    try {
                        sh 'docker stop $(docker ps -q --filter ancestor=backend:latest)'
                        sh 'docker rm $(docker ps -a -q --filter ancestor=backend:latest)'
                    } catch (Exception e) {
                        echo "An error occurred: ${e}"
                    }
                }
            }

            post {
            success {
                  echo 'Stop and Remove success!'
            }
          }
        }


        stage('Bulid Backend') {
          // 도커 빌드
          agent any
          steps {
            echo 'Build Backend'

            // 도커 이미지 생성
            // server라는 이름으로 빌드, 도커파일이 현재 경로에 있어서 . 써줌
            // docker build . -t server --build-arg env=${PROD}
            // -t 옵션으로 이미지에 server라는 태그를 달아줌
            dir ('./Backend'){
                sh """
                docker build . -t backend:latest
                """
            }
          }

          post {
            // steps 끝나면 post온다
            // 빌드하다 실패하면 error 뱉고, 나머지 과정 하지말고 종료
            failure {
              error 'This pipeline stops here...'
            }
          }
        }


        stage('Deploy Backend') {
          agent any

          steps {
            echo 'Build Backend'
            // 위에서 만든 이미지를 실행시키기 (원래 떠있던 이미지 지우고 실행)
            // 기존에 돌리던 이미지가 있을때만 run 코드 전에 docker rm -f $(docker ps -aq) 추가
            // server라는 태그
            dir ('./Backend'){
                sh '''
                docker run -p 9000:9000 -e TZ=Asia/Seoul --env-file /home/ubuntu/myEnv/springEnv --network ssafy-network -d backer
                '''
            }
          }

          post {
            success {
                  echo 'Deploy Backend success!'
            }
          }
        }

    }
}
```

- Backend Docker file 작성

```
# FROM은 베이스 이미지(Base image)를 지정
FROM adoptopenjdk:11-hotspot

COPY gradlew .
COPY gradle gradle
```

```
COPY build.gradle .
COPY settings.gradle .
COPY src src

# RUN 은 말 그대로 command 를 실행(run)하여 새 이미지에 포함시키는 역할

RUN chmod +x ./gradlew
# 실행 가능한 jar파일 생성
RUN ./gradlew bootJar

# base image
FROM adoptopenjdk:11-hotspot
# builder 이미지에서 build/libs/*.jar 파일을 app.jar로 복사
COPY --from=0 build/libs/*.jar app.jar

EXPOSE 9000
ENTRYPOINT ["java","-jar", "/app.jar"]
```

- Frontend Docker file 작성

```
# Use an official Node.js runtime as the base image
FROM node:18

# Set the working directory in the container
WORKDIR /app

# Copy the package.json and package-lock.json files
COPY package*.json ./

# Install the dependencies
RUN npm install

# Copy the React project files
COPY . .

# Build the React project
RUN npm run build

# Use Nginx as the web server
FROM nginx

# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d

# Copy the build files to the Nginx web root directory
COPY --from=0 /app/build /usr/share/nginx/html

# Expose port 80
EXPOSE 80

# Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

- jenkins 크레덴셜에 gitlab key 등록 (디플로이 토큰으로)
- new item - pipeline 선택하여 구성에 pipeline script에 젠킨스 파이프라인 스크립트 붙여넣기
- 지금빌드 클릭

II. 외부 서비스

1. 소셜 로그인

1. https://developers.kakao.com/console/app 앱등록
    a. key 발급
        - 네이티브 앱 키: Kakao SDK for Android 초기화, Kakao SDK for iOS 초기화 시 사용

- REST API 키: REST API 요청 시 HTTP 헤더(Header)에 전달
- JavaScript 키: Kakao SDK for JavaScript 초기화 시 사용
- Admin 키: 일부 관리자 기능에 사용, 모든 권한을 갖고 있는 키이므로 유출되지 않도록 주의

2. 플랫폼 등록
   a. 도메인 주소

```
http://localhost:3000/oauth/kakao
http://i8a308.p.ssafy.io/oauth/kakao
https://localhost:3000/oauth/kakao
https://i8a308.p.ssafy.io/oauth/kakao
```

   b. OpenID Connect 활성화 설정
   c. 카카오 로그인 활성화 설정
   d. Redirect Url http://localhost:3000/oauth/kakao/callback 설정(본인 서비스에 설정한 URL으로 설정)
   - 동의항목 설정가능 (받아올 정보)

3. REST_API_KEY 와 REDIRECT_URI 는 따로 env파일을 만들어서 관리

```
REACT_APP_KAKAO_REST_API_KEY = '204f458585e0229e8443cd7bc1be5c5e'
REACT_APP_KAKAO_REDIRECT_URI ='http://localhost:3000/oauth/kakao'
REACT_APP_REST_DEFAULT_URL = 'http://localhost:9000/api'
```

인가코드는 백엔드와 통신해서 전달한다.