포팅 메뉴얼_A708

1. Stacks
    Issue Management
    SCM
    Communities
    Infra
    `AWS`
    `Front-end server`
    `Back-end server`
    `DataBase server`
    `Jenkins server`
2. Build & Distribute
3. EC2 setting command
    Docker 설치
    Database 컨테이너 실행
    Jenkins 세팅
Dashboard → 새로운 item생성
    Backend project(2개 prod, dev)
    Frontend project(1개 front)
4. Database Dump 파일 MariaDB Container 주입
5. Local Test command
    Front-end Local build
    Back-end Local build
6. Files ignored
7. Social Login
    Kakao Login
8. dev server → prod server 전환 방법

1. Stacks

Issue Management

- JIRA
- GITLAB

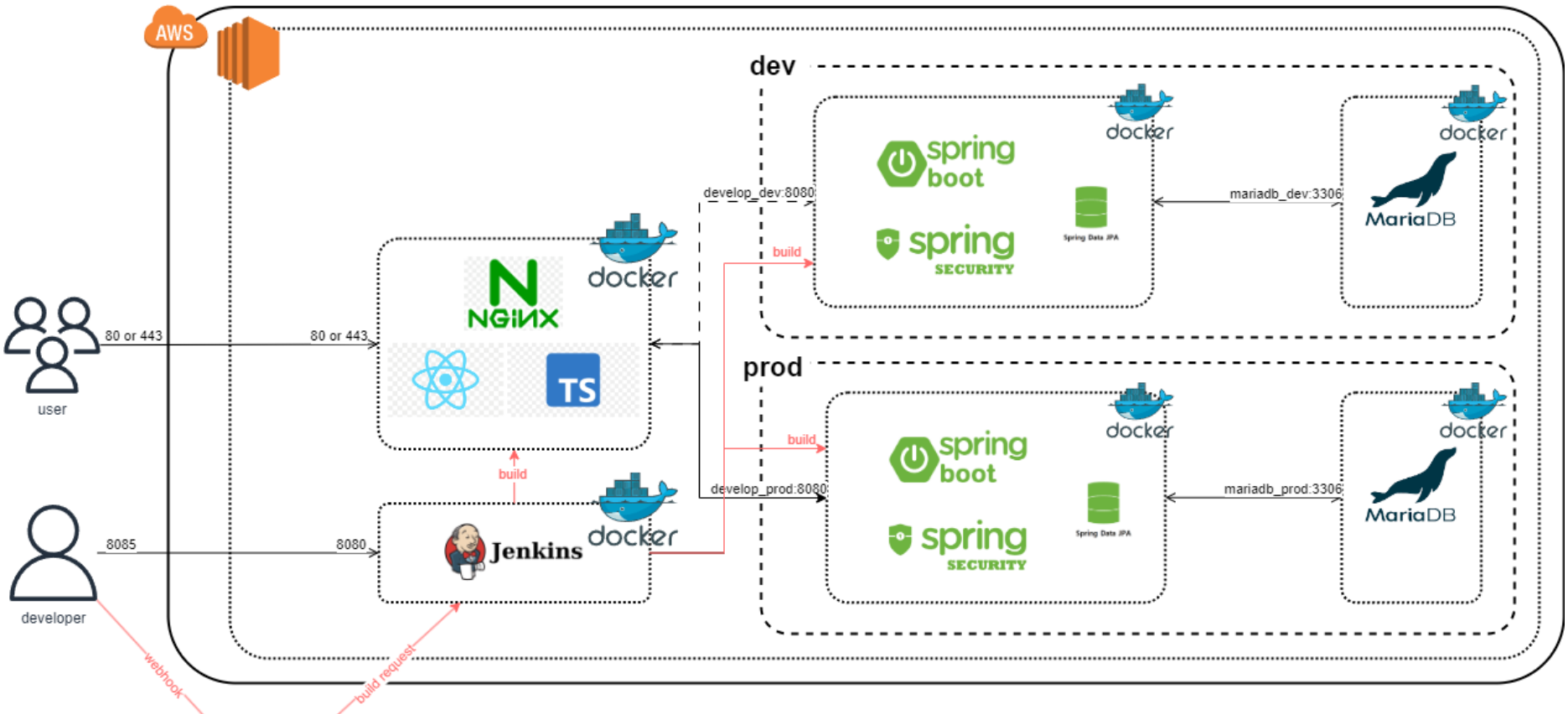
SCM

- GITLAB

Communities

- Mattermost
- Notion

Infra



`Design`

- Figma

`AWS`

- EC2 (ubuntu 20.04 LTS)

`Front-end server`

- NGINX - stable-alpine ver. (**외부포트: 80, 443, 내부포트: 80, 443**)
- **React + Typescript**
  - React - 18.2.0 ver.
  - TypeScript - 5.1.6 ver.
  - Node - 18.16.1 ver.
- **Style**
  - Material UI + styled-component
    - MUI - 5.14.0 ver.
    - styled-component - 6.0.0 ver.
- **Linting**
  - eslint, prettier
    - eslint - 8.38.0 ver.
- **Bundling**
  - Vite - 4.4.7 ver.
- **Etc**
  - axios - 1.4.0 ver.
  - react-router-dom - 6.14.1 ver.
  - chart library
    - recharts - 2.7.2 ver.
  - carousel library
    - swiper - 10.1.0 ver.
    - react-slick - 0.29.0 ver.
    - react-items-carousel - 2.8.0 ver.

`Back-end server`

- gradle 8.1.1
- develop 서버 (**내부포트: docker network, 외부포트 : 8080**)

- prodution 서버 (**내부포트: docker network, 외부포트 : 8080**)
- spring boot - 2.7.13 ver.
- jdk - 11 ver.
- querydsl - querydsl-jap:5.0.0 ver.
- JWT - io.jsonwebtoken:jjwt 0.9.1 ver.
- Spring Security - spring boot 버전과 동일
- JPA - spring boot 버전과 동일

`DataBase server`

- MariaDB - 11.0.2 ver.
- MariaDB - Dev **(포트 : 3306)**
- MariaDB - Prod **(포트 : 3306)**

`Jenkins server`

- Jenkins - latest ver. **(외부 포트 : 8085, 내부포트: 8080)**

2. Build & Distribute

- SpringBoot
- Nginx
- Jenkins
- MariaDB

3. EC2 setting command

- EC2 접속

  pem파일이 있는 폴더에서 아래 커맨드 실행

  ```
  ssh -i I9A708T.pem ubuntu@i9a708.p.ssafy.io
  ```

  > 💡 pem 파일 → 모드 400

- 포트 설정

  ```
  ufw allow 22 # 우분투 방화벽
  ufw allow 80 # 웹 기본 포트
  ufw allow 443 # https 포트
  ufw allow 8085 # jenkins 포트
  ufw enable
  ufw status numbered # 허용된 포트 확인
  ```

Docker 설치

- Docker 설치

  ```
  apt update
  apt install docker
  ```

- Docker network 생성

  ```
  docker network create 8low_network
  ```

Database 컨테이너 실행

- mariadb_dev 컨테이너 실행 → {비밀번호}부분 바꿔줄 것

  ```
  docker run --name mariadb_dev -d -v /home/ubuntu/mariadb_dev:/var/lib/mysql --restart=always --network 8low_network -e MYSQL_ROOT_PASSWORD={비밀번호} mariadb:11.0.2
  ```

- mariadb_prod 컨테이너 실행 → {비밀번호}부분 바꿔줄 것

  ```
  docker run --name mariadb_prod -d -v /home/ubuntu/mariadb_prod:/var/lib/mysql --restart=always --network 8low_network -e MYSQL_ROOT_PASSWORD=ssafy1234 mariadb:11.0.2
  ```

Jenkins 세팅

- Jenkins 컨테이너 실행

```
sudo docker run -d --name jenkins --restart=on-failure \
-p 8085:8080 \
-v /var/jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-e TZ=Asia/Seoul \
-u root \
--privileged \
jenkins/jenkins
```

- i9a708.p.ssafy.io:8085 접속
- 초기 비밀번호 확인

```
docker logs jenkins -f
```

- 초기 비밀번호 입력
- Dashboard > Jenkins 관리 > Plugins 플러그인 설치
  - GitLAB
  - Docker
  - Publish Over SSH
- Dashboard > Jenkins 관리 > System
  - Publish over SSH 설정
    - key → EC2 privatekey 추가
    - SSH Server
      - name → EC2 별명 추가 (ex: SSAFY Common Project EC2)
      - Hostname → i9a708.p.ssafy.io
      - Username → root (사용하려면 EC2에 root 권한 추가)
- Dashboard > Jenkins 관리 > Credentials
  - GitLabToken추가
    - Kind : Username with password
      - username : GitLab 아이디
      - Password : GitLab에서 발급한 ACCESS TOKEN
  - DATABASE_URL_DEV
    - Kind : Secret text
    - Secret : `jdbc:mariadb://mariadb_dev:3306/decalcomanie?useSSL=false&serverTimezone=Asia/Seoul&allowPublicKeyRetrieval=true`
  - DATABASE_USERNAME_DEV
    - Kind : Secret text
    - Secret : DEV 데이터베이스 아이디
  - DATABASE_PASSWORD_DEV
    - Kind : Secret text
    - Secret : DEV 데이터베이스 비밀번호
  - DATABASE_URL_PROD
    - Kind : Secret text
    - Secret : `jdbc:mariadb://mariadb_prod:3306/decalcomanie?useSSL=false&serverTimezone=Asia/Seoul&allowPublicKeyRetrieval=true`
  - DATABASE_USERNAME_PROD
    - Kind : Secret text
    - Secret : PROD 데이터베이스 아이디
  - DATABASE_PASSWORD_PROD
    - Kind : Secret text
    - Secret : PROD 데이터베이스 비밀번호
  - JWT_SECRET_KEY
    - Kind : Secret text
    - Secret: 암호화 한 128비트 이상의 값
  - KAKAO_RESTAPI_KEY
    - Kind : Secret text
    - Secret: 카카오 developer에서 발급 받은 RESTAPI KEY

- KAKAO_CLIENT_ID
  - Kind : Secret text
  - Secret: 카카오 developer에서 발급 받은 RESTAPI KEY
- KAKAO_CLIENT_SECRET
  - Kind : Secret text
  - Secret: 카카오 developer 로그인 제품 설정에서 발급 받은 Client Secret
- KAKAO_REDIRECT_URL
  - Kind : Secret text
  - Secret: kakao developer에서 설정한 redirect url
- KAKAO_ADMIN_KEY
  - Kind : Secret text
  - Secret: kakao developer에서 발급 받은 Admin Key

Dashboard → 새로운 item생성

Backend project(2개 prod, dev)

- project 이름 설정(Freestyle project) : 관리하고 싶은 컨테이너
  - 소스 코드 관리
    - Git
      - Repository URL : GitLab https clone URL
      - Credentials : GitLabToken
    - Branch Specifier : 접근하고 싶은 branch명 (ex: */master)
- 빌드 유발 설정
  - Build when a change is pushed to GitLab. GitLab webhook URL : `{webhook URL}` 체크

    webhook URL을 복사해서 GitLab Repository > Setting > Webhooks URL에 넣고 Trigger를 Push events와 Merge request events를 선택
    - 고급 탭

      Secret token을 발급하고 GitLab Webhooks Secret token 설정에 추가

    GitLab Webhook에서 Add webhook 버튼으로 추가
- 빌드 환경
  - Use secret text or file
    - Secret text 추가
      - DATABASE_URL
      - DATABASE_USERNAME
      - DATABASE_PASSWORD
      - JWT_SECRET_KEY
      - KAKAO_CLIENT_ID
      - KAKAO_CLIENT_SECRET
      - KAKAO_REDIRECT_URL
      - KAKAO_ADMIN_KEY
- 빌드 스탭
  - Execute shell

```
cd ./server/src/main/resources;
echo "DATABASE_URL: $DATABASE_URL\n\
DATABASE_USERNAME: $DATABASE_USERNAME\n\
DATABASE_PASSWORD: $DATABASE_PASSWORD\n\
JWT_SECRET_KEY: $JWT_SECRET_KEY\n\
KAKAO_CLIENT_ID: $KAKAO_CLIENT_ID\n\
KAKAO_CLIENT_SECRET: $KAKAO_CLIENT_SECRET\n\
ACCESS_TOKEN_EXPIRE: 180\n\
REFRESH_TOKEN_EXPIRE: 604800\n\
KAKAO_REDIRECT_URL: $KAKAO_REDIRECT_URL\n\
KAKAO_ADMIN_KEY: $KAKAO_ADMIN_KEY" > "env.yml";
```

  - Invoke Gradle script
    - Invoke Gradle

      Gradle Version : Gradle 8.1.1
    - Task

```
clean build -x test
```

- Send files or excute commands over SSH

  SSH Publishers

  SSH Server

  - Name : EC2 별명

- 빌드 후 조치
  - **Send build artifacts over SSH**

    SSH Publishers

    SSH Server

    - Name : EC2 별명

    고급

    Transfer Set → project이름 및 container 이름 세팅

    Source files : `project이름/server/build/libs/*.jar`

    Remove prefix : `project이름/server/build/libs/`

    Exec command

```
docker stop {dockercontainer이름}
docker rm {dockercontainer이름}
docker rmi {dockerimage이름}
docker build -t {dockerimage이름} /var/jenkins_home/workspace/{project이름}/server
docker run -d --name {dockercontainer이름} --network 8low_network {dockerimage이름}
```

Frontend project(1개 front)

⚠️ EC2 /front_prod/etc에 fullchain1.pem과 privkey1.pem을 certbot을 통해 인증서를 발급할 것!

- project 이름 설정(Freestyle project)
  - 소스 코드 관리
    - Git
      - Repository URL : GitLab https clone URL
      - Credentials : GitLabToken
    - Branch Specifier : 접근하고 싶은 branch명 (ex: */master)
- project 이름 설정(Freestyle project) : 관리하고 싶은 컨테이너
  - 소스 코드 관리
    - Git
      - Repository URL : GitLab https clone URL
      - Credentials : GitLabToken
    - Branch Specifier : 접근하고 싶은 branch명 (ex: */master)
- 빌드 환경
  - Delete workspace before build starts 체크
  - Use secret text(s) or file(s) 체크
    - Secret text
      - KAKAO_RESTAPI_KEY
- 빌드 스텝
  - Execute shell

```
cd ./client;
echo "VITE_REACT_APP_HOST='http://localhost:5173'\n\
VITE_REACT_APP_SERVER='http://localhost:8080'\n\
VITE_REACT_APP_APIKEY='$KAKAO_RESTAPI_KEY'" > ".env.development"
echo "VITE_REACT_APP_HOST='https://decalcomanie.site'\n\
VITE_REACT_APP_SERVER='https://decalcomanie.site/api'\n\
VITE_REACT_APP_APIKEY='$KAKAO_RESTAPI_KEY'" > ".env.production"
```

- 빌드 후 조치
  - **Send build artifacts over SSH**

    SSH Publishers

SSH Server

- Name : EC2 별명

고급

Transfer Set → project이름 및 container 이름 세팅

Exec command

```
docker stop {dockercontainer이름};
docker rm {dockercontainer이름};
docker rmi {dockerimage 이름};
docker build -t {dockerimage 이름} /var/jenkins_home/workspace/{project이름}/client;
docker run -d --name {dockercontainer이름} --network 8low_network -v /front_prod/cert:/etc/letsencrypt -p 443:443 -p 80:80 {dockerimage 이름};
```

4. Database Dump 파일 MariaDB Container 주입

- EC2에 sql파일 가져오기

```
scp -i {EC2 인스턴스에 연결할 때 사용되는 개인 키 파일의 경로} {로컬에서 복사하려는 파일의 경로} {대상 EC2 인스턴스의 공용 IP 주소나 DNS 이름}:{EC2 인스턴스 내에서 파일을 복사할 경로}
```

- MariaDB Conatiner 내부로 sql파일 복사

```
docker cp {폴더경로/파일명.sql} {컨테이너이름:/usr/bin}
```

- MariaDB Container 접속

```
docker exec -it {컨테이너이름} /bin/sh
```

- MariaDB 접속

```
mariadb -u {mariadb아이디} -p
{mariadb비밀번호}
```

- Source 실행

```
source {파일명.sql}
```

5. Local Test command

Front-end Local build

- React App 으로 이동 (**S09P12A708/client)**
- 아래 명령어로 실행

```
npm run dev
```

Back-end Local build

- Back-end project로 이동 (**S09P12A708/server)**
- Intellij로 열기
  - File > Project structure > Project Settings > Project > SDK > 11 version
  - File > Project structure > Project Settings > Modules > Module SDK > 11 version
  - File > Settings > Build, Excution, Deployment > Build Tools > Gradle > Gradle JVM > 11 version
- **server/src/main/resources/env.yml 생성**

```
DATABASE_URL: jdbc:mariadb://localhost:3305/decalcomanie?useSSL=false&serverTimezone=Asia/Seoul&allowPublicKeyRetrieval=true
DATABASE_USERNAME: {localDB 아이디}
DATABASE_PASSWORD: {localDB 비밀번호}
JWT_SECRET_KEY: {512bits 이상의 암호화된 문자열}
KAKAO_CLIENT_ID: {kakao developer restapi key}
KAKAO_CLIENT_SECRET: {kakao developer client secret key}
ACCESS_TOKEN_EXPIRE: 180
REFRESH_TOKEN_EXPIRE: 604800
KAKAO_REDIRECT_URL: {kakao developer에 등록한 http://localhost:5173/oauth/kakao/callback}
KAKAO_ADMIN_KEY: {kakao developer admin key}
```

- **server/src/main/resources/application.yml 수정**

  5번째 줄 주석 → context-path: /api

- **server/src/main/java/com/eightlow/decalcomanie/DecalcomanieApplication.java 실행**

6. Files ignored

- Backend
  - server/src/main/resources/env.yml
- Frontend
  - client/.env.development
  - client/.env.production

7. Social Login

Kakao Login

- kakao key setting
  - front, back
    - front : `/client/.env.development` 파일에 작성(local의 경우)
    - back : `server/src/main/resources/env.yml` 파일에 작성(local의 경우)
    - jenkins credential secret text 세팅(ec2의 경우)
      - KAKAO_RESTAPI_KEY → KAKAO RESTAPI KEY
      - KAKAO_CLIENT_ID → KAKAO RESTAPI KEY
      - KAKAO_CLIENT_SECRET → KAKAO CLIENT SECRET
      - KAKAO_REDIRECT_URL → KAKAO REDIRECT URL
      - KAKAO_ADMIN_KEY → KAKAO ADMIN KEY
    - `decalcomanie` KAKAO RESTAPI KEY : {KAKAO DEVLEOPERS 발급}
    - `decalcomanie` KAKAO CLIENT SECRET : {KAKAO DEVELOPERS 발급}
    - `decalcomanie` KAKAO ADMIN KEY : {KAKAO DEVELOPERS 발급}

8. dev server → prod server 전환 방법

- client/nginx/nginx.conf 코드 변경

  `http://develop_dev` → `http://develop_prod`