SAMSUNG SW ACADEMY FOR YOUTH

對外秘

프로젝트 명세서

도커 실습

對外秘

목차

對外秘

1. 프로젝트 개요

본 프로젝트는 많은 기업에서 사용하는 도커라는 컨테이너 기술을 실습과 함께 기본적인 개념 및 명령어를 학습하고 실제 공통 프로젝트 결과물을 도커화(Dockerize) 해보는 과정입니다. 실습 및 과제를 마치고 내용을 잘 정리해 보고 이후 스스로 어떻게 도커를 활용 가능할지 한번 생각해 보세요. 과제를 진행함에 있어 이미 도커를 잘 알고 있는 교육생은 심화(추가) 과제까지 진행해 보기를 추천 드립니다.

여러분이 로컬에서 개발하던 환경을 AWS 에서 배포하면서 다양한 언어별 환경구성, 의존성 패키지/라이브러리, 빌드 툴 등등 설치할 때 어려움을 겪었을 것입니다. 매번 새로운 버전이 출시 되어 버전이슈도 발생하고 생각지도 못한 환경적인 문제로 여러분의 코드가 새로운 배포 환경에서 예상처럼 동작해 주는 것을 보장받기 힘듭니다. 또한 이를 수동으로 매번 여러 대의 서버에 세팅하는 것은 작업자의 실수를 야기할 수도 있고 배포 경험은 작업자에게 국한되어 공유되기 힘듭니다. 도커는 이를 효율적으로 개선시켜 줍니다. OS 를 포함한 설치과정은 Dockerfile 로 문서화 되고 수정 이력은 버전관리가 되어 변경사항을 쉽게 확인 가능합니다. 그래서 문제 발생시 언제든 롤백 하기도 편리하고 VM 이미지 대비 용량 및 실행속도가 월등히 빠릅니다. 왜 개발자가 도커를 알아야 할까요? 요즘은 DevOps 문화가 확산되어 인프라를 잘 모르는 개발자도 쉽게 서버를 제작할 수 있고 개발하면서 자주 접할 가능성이 높습니다. 또한 도커를 사용하면 개발과정에서 필요한 환경 구성이 편리해 지고 여러 가지 서비스를 레고블럭 처럼 쌓아서 매쉬업하기도 용이합니다. 도커로 여러분의 서비스를 윈도우/맥/리눅스 가리지 않고 동일하게 동작함을 보장하는 불변서버로 이미지를 굽는 것처럼 제작해 보세요. 이러한 장점으로 우리가 알고 있는 많은 서비스들이 이미 도커 이미지를 제공하고 있는 것처럼 이는 선택이 아닌 필수로 자리잡아 가고 있습니다. 도커 허브(https://hub.docker.com/search)를 방문해 보시고 필요한 서비스는 docker run 명령어를 사용하여 컨테이너로 손쉽게 구동해 보세요. 도커의 세계에 오신 것을 환영합니다.

2. 과제

공통 웹모바일 프로젝트 트랙 2(웹+디자인) Skeleton 코드를 기반으로 도커 이미지를 제작하고 실행하는 과제입니다. 처음 도커를 접하는 교육생을 대상으로 하였으며 이미 도커를 잘 알고 있는 교육생은 심화(추가) 과제까지 진행해 보기를 추천 드립니다. 과제는 다음과 같은 크게 3 가지 순서로 진행됩니다.

1) 설치 및 명령어 실습

도커를 설치하여 기본적인 이미지 및 컨테이너 생성 방법을 실습합니다.

2) 여러가지 서비스 도커로 이용해 보기

도커 허브에 공개된 수 많은 유용한 이미지들 중에 샘플로 Jenkins 를 docker run 명령어 한 줄로 쉽게 구축해 봅니다.

3) Skeleton 코드로 도커 이미지 제작

공통 웹모바일 프로젝트의 트랙 2(웹+디자인) Skeleton 코드를 샘플로 프론트엔드와 백엔드 도커 이미지를 제작해 봅니다.

※ 참고자료

도커를 학습하기 위해서 구글에서 검색되는 자료들을 참고하면 됩니다. 아래 참고 자료 또한 다양한 자료들 중 하나의 예이므로 공식 문서와 잘 설명된 예시들을 찾아 살펴보는 시간을 아끼지 말고 꼭 여러분의 지식과 경험으로 만드시기 바랍니다

구분	제목	링크
지초 자료	도커 튜토리얼	https://www.44bits.io/ko/post/easy-deploy-with-docker
심화 자료	도커 컴포즈를 활용	https://www.44bits.io/ko/post/almost-perfect-development-environment-with-docker-and-docker-compose
동영상 강의	생활코딩 도커	https://opentutorials.org/course/128/8657

※ 전체 과제 목록

Req.	Category
2.1	도커 개발환경 설정
2.2	도커 기본 명령어 실습
2.3	샘플 서비스(Jenkins)를 이용한 도커 실습
2.4	Skeleton 코드 기반 도커 이미지 제작 (프론트)
2.5	Skeleton 코드 기반 도커 이미지 제작 (백엔드)

2.1 도커 개발환경 설정

1. 도커 설치: 사이트에서 OS 에 맞는 버전 설치

https://www.docker.com/products/docker-desktop

2. 설치 후 도커 버전 확인

```
docker -v
```

3. 테스트용 Hello world 도커 컨테이너 실행

```
docker run hello-world
```

※ hello world 도커 이미지가 자동 다운로드되어 실행된 결과로 "Hello from Docker!" 가 포함된 메시지가 출력되면 성공

※ Tip: Hello world 도커 이미지만 다운로드 하려면 docker pull hello-world 실행

2.2 도커 기본 명령어 실습

1. 컨테이너 조회

```
docker ps -a
```

※ Tip: -a 옵션은 정지(실행 종료)된 컨테이너까지 조회하는 옵션

2. Hello world 컨테이너 삭제

```
docker rm [컨테이너 ID 또는 NAME]
```

※ Tip: 컨테이너 실행 시 docker run -name=[컨테이너 이름] ~ 과 같이 --name 옵션을 추가하면 자동으로 만들어진 이름 대신 지정된 이름으로 컨테이너 생성 가능

※ Tip: 컨테이너 ID 는 앞 일부분만 입력 가능 ex) docker rm 76

3. 도커 이미지 조회

```
docker images
```

4. Hello world 도커 이미지 삭제

```
docker rmi [이미지 ID 또는 이미지명:TAG 명]
```

※ Tip: TAG 중에 최신을 의미하는 latest 태그명은 생략 가능

2.3 샘플 서비스(Jenkins)를 이용한 도커 실습

1. Jenkins 를 도커 컨테이너로 실행 및 실행(Up STATUS)중인지 확인

```
docker run --name myjenkins -d -p 8080:8080 jenkins:2.60.3

docker ps
```

※ 참고: docker hub 사이트 방문: https://hub.docker.com/_/jenkins

※ Tip: -d 옵션은 백그라운드 데몬으로 실행 시키는 옵션이고

-p 는 가상 머신(Guest PC)에서 동작하는 서비스에 docker proxy 서비스가 NAT 기능을 수행해 포트 포워딩을 해주어 실제 머신(HOST PC) IP 로 서비스에 접속이 가능하도록 해준다.

※ Tip: 컨테이너가 삭제되면 생성된 데이터가 같이 삭제되므로 보통 호스트 PC 의 디렉토리에 데이터가 생성되도록 컨테이너에는 볼륨 마운트 옵션을 추가하여 실행시킵니다. 추가적인 옵션: -v /your/home:/var/jenkins_home

2. Jenkins 서버 컨테이너의 bash 실행 후 컨테이너의 OS 버전 확인

```
docker exec -it myjenkins bash

jenkins@92552ea0b836:/$ cat /etc/issue
Debian GNU/Linux 9 \n \l
```

※ 참고: Jenkins 컨테이너는 bash 가 포함되어 있지만 없는 서비스의 경우에는 bash 대신 sh 를 사용해도 된다.

※ 참고: 윈도우 git-bash 에서 "the input device is not a TTY." 에러가 발생하면 명령 프롬프트(cmd) 창 이용

※ 참고: 컨테이너의 bash 가 실행된 상태는 마치 ssh 로 리눅스 서버에 접속해서 명령어를 사용하는 것과 비슷하다.

※ Tip: -it 옵션은 -i: interactive, -t: tty 가 합쳐진 의미로 터미널에서 입출력이 가능해진다.

3. 컨테이너 안(bash)에서 Admin 패스워드가 저장된 파일 확인 후 컨테이너 bash 종료

```
jenkins@92552ea0b836:/$ cat /var/jenkins_home/secrets/initialAdminPassword
[패스워드가 출력됨]
jenkins@92552ea0b836:/$ exit
```

4. 2-3 으로 두 번의 과정으로 패스워드를 확인했던 과정을 한번에 실행하기

```
docker exec myjenkins cat /var/jenkins_home/secrets/initialAdminPassword
[패스워드가 출력됨]
```

※참고: docker 명령어는 개발 PC(Host PC)에서 실행하는 명령어 이므로 컨테이너 안쪽(bash)에서 실행하는 방식과 비교를 위해 설명한 것이므로 실행 위치를 혼동되지 않도록 실습해야 합니다.

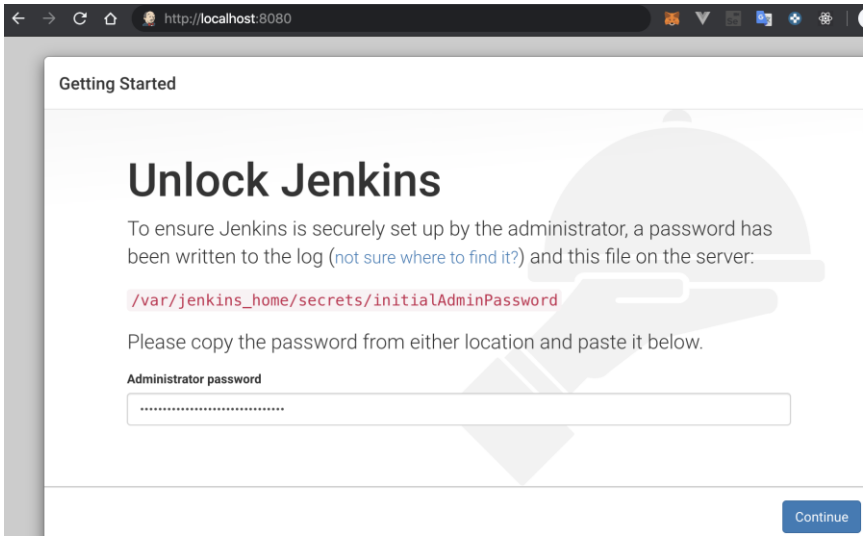
※Tip: docker logs myjenkis 명령어를 실행해서 컨테이너 실행 시 출력된 메시지에서도 패스워드 확인이 가능합니다.

5. 컨테이너 안에 있는 패스워드 파일을 개발 PC 로 복사하기

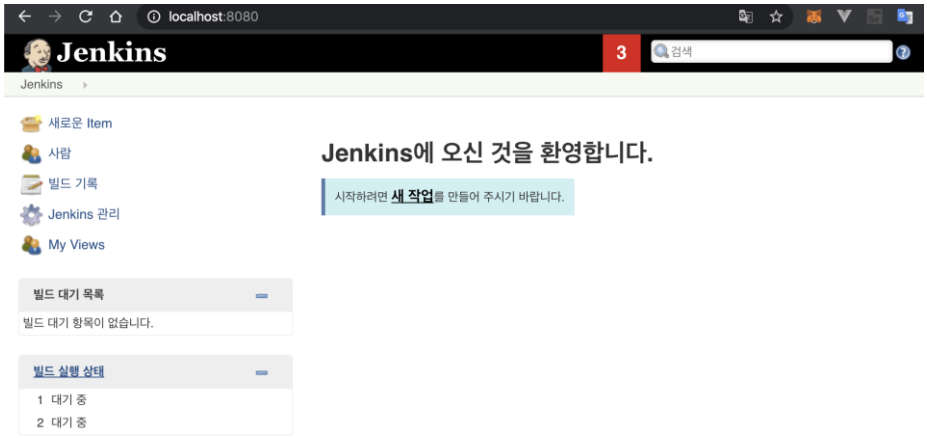
```
docker cp myjenkins:/var/jenkins_home/secrets/initialAdminPassword ./

dir
initialAdminPassword
```

※Tip: 이처럼 컨테이너 안에 있는 패스워드 파일을 내 개발PC로 복사해서 확인해 볼 수도 있다.
반대로 컨테이너 안에 파일을 복사해 넣으려면 다음 명령어로 가능하다. (scp 명령 방식과도 비슷)
docker cp 파일명(개발PC에서 복사 시킬 파일) myjenkins:/var/jenkins_home(컨테이너 경로)

6. 웹 브라우저에서 접속(http://localhost:8080)해서 앞서 확인 한 패스워드를 붙여 넣고 Jenkins 설정을 계속 진행하기



7. 계정 설정까지 완료해서 환영 페이지 확인



8. 컨테이너를 재 시작 해서 Up Status 시간이 초기화 된 것을 확인 후 삭제

```
docker restart myjenkins
docker ps

docker rm -f myjenkins
```

※ Tip: docker start/stop/restart 를 사용하면 마치 서비스를 시작/정지/재 시작 하는 것처럼 컨테이너 레벨로 서비스 제어가 가능하다.

※ Tip: -f 옵션은 실행중인 컨테이너도 강제 삭제 가능

※ 참고: 앞에서 배운 이미지 삭제하는 명령어를 통해 Jenkins 이미지도 정리해 보자

2.4 Skeleton 코드 기반 도커 이미지 제작 (프론트)

1. 공통 웹/모바일 트랙 2(웹+디자인) skeleton 코드 다운로드

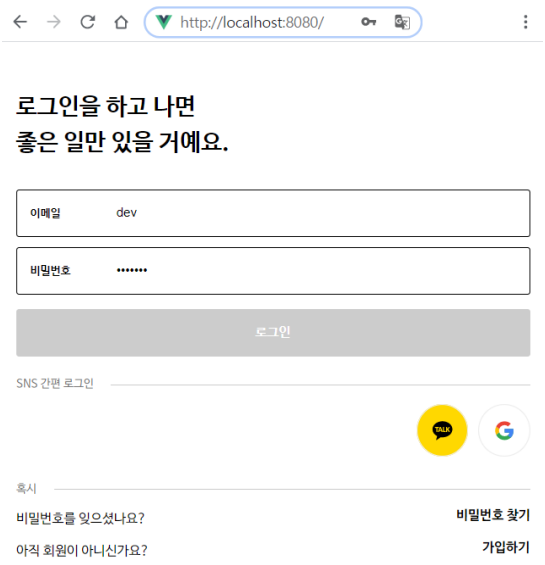
```
git clone https://lab.ssafy.com/ssafyadmin/webmobile2-skeleton.git

cd webmobile2-skeleton

ls
back-sk   front-sk  wireframe
```

2. 로컬에서 프론트 실행 및 웹 브라우저로 접속(http://localhost:8080)해서 확인

```
cd front-sk
yarn install
yarn serve
```

※ *참고: Node.js 및 yarn 은 설치되어 있다고 가정함 - 설치관련 내용은 SSAFY GIT 명세서에 있음*

※ *참고: 간단히 정상 접속 됨을 확인했다면 이후 실습을 위해 서버는 종료합니다.*



3. webmobile2-skeleton/front-sk/Dockerfile 을 작성 후 프론트 코드용 도커 이미지 빌드 (Dockerfile 참고: https://kr.vuejs.org/v2/cookbook/dockerize-vuejs-app.html)

```
dokcer build . -t front:0.1
…
Successfully built b35805335785
Successfully tagged front:0.1


docker images
REPOSITORY    TAG       IMAGE ID          CREATED           SIZE
front         0.1       b35805335785      39 hours ago      22.7MB
```

※ *Tip: -t 는 도커 이미지 TAG 명으로 보통 버전관리 용도로 사용한다.*

※ Tip: 도커 이미지 빌드시 sh: vue-cli-service: not found 가 발생하면 아래와 같은 내용을 추가

RUN npm install --production

RUN npm install @vue/cli-service <- 필요

4. 이미지에 TAG 추가하기

```
docker tag front:0.1 front:latest


docker images
REPOSITORY    TAG       IMAGE ID          CREATED           SIZE
front         latest    b35805335785      39 hours ago      22.7MB
front         0.1       b35805335785      39 hours ago      22.7MB
```

※ Tip: TAG는 이미지 ID에 대한 alias/포인터와 같이 이미지 이름처럼 사용해서 쉽게 관리하는데 도움이 된다.

5. 이미지에 TAG 삭제하기

```
docker rmi front
Untagged: front:latest


docker images
REPOSITORY    TAG       IMAGE ID          CREATED           SIZE
front         0.1       b35805335785      39 hours ago      22.7MB
```

※ Tip: 도커 이미지 삭제하는 명령어로 TAG를 삭제하고 실제 이미지는 모든 TAG가 삭제될 때 삭제된다.

6. 도커로 프론트 실행 및 웹 브라우저로 접속(http://localhost)해서 확인

```
docker run -it -p 80:80 --rm front:0.1
[실행해도 아무런 로그가 출력되지 않고 브라우져로 접속시 access log 출력됨]
```

※ Tip: --rm은 컨테이너가 정지되면 자동으로 삭제되는 옵션이고

임시테스트용으로 컨테이너를 실행시켜 보는 용도로 유용하다.

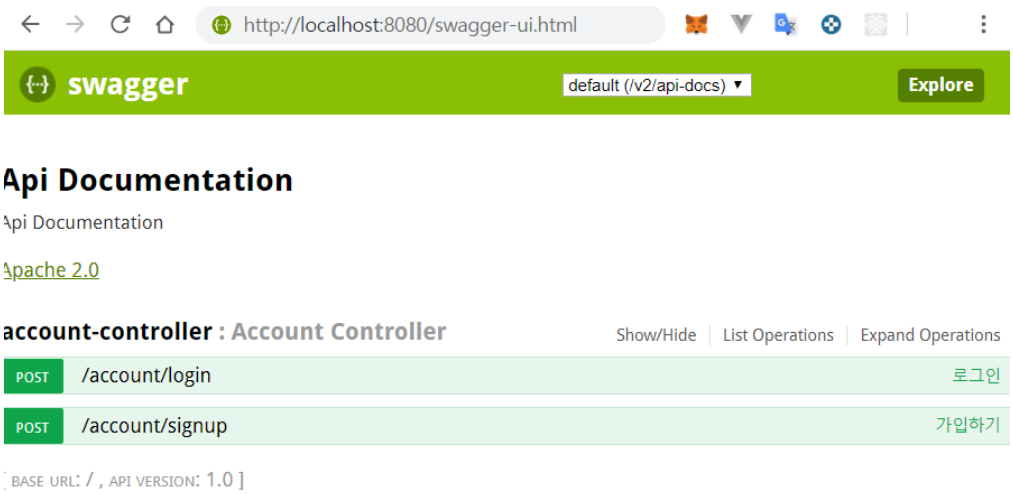
※ Tip: 컨테이너가 포그라운드로 실행 중이므로 "Ctrl + C" 키 입력으로 종료한다.

2.5 Skeleton 코드 기반 도커 이미지 제작 (백엔드)

1. 로컬에서 백엔드 실행 및 확인(http://localhost:8080/swagger-ui.html)

```
cd back-sk
mvnw package
java -jar target\webcuration-0.0.1-SNAPSHOT.jar
```

※ 참고: Java 는 설치되어 있어야 하고, 간단히 정상 접속 됨을 확인했다면 이후 실습을 위해 서버는 종료합니다.



2. webmobile2-skeleton/back-sk/Dockerfile 을 작성 완료 후 백엔드 코드용 도커 이미지 빌드

```
dokcer build . -t back:0.1
…
Successfully built 71d57b43c6ec
Successfully tagged back:0.1

docker images
REPOSITORY    TAG       IMAGE ID          CREATED           SIZE
back          0.1       71d57b43c6ec      39 hours ago      134MB
front         0.1       b35805335785      39 hours ago      22.7MB
```

※ 참고: https://spring.io/guides/gs/spring-boot-docker

3. 도커로 백엔드 실행 및 확인(http://localhost:8080/swagger-ui.html)

```
docker run -it -p 8080:8080 --rm back:0.1
```

※ Tip: --rm 은 컨테이너가 정지되면 자동으로 삭제되는 옵션이고
임시테스트용으로 컨테이너를 실행시켜 보는 용도로 유용하다.

4. 개발용 DB 로 사용할 MySQL 컨테이너 실행 및 정상 실행 여부 조회(docker ps)

```
docker run --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=ssafyssafyroomroom -e
MYSQL_DATABASE=ssafy -d mysql --character-set-server=utf8mb4 --collation-
server=utf8mb4_unicode_ci

docker ps
CONTAINER ID   IMAGE  COMMAND                  PORTS                    NAMES
b3430d9132dc   mysql  "docker-entrypoint.s…" 0.0.0.0:3306->3306/tcp mysql
```

※ 참고: -e 옵션은 컨테이너 OS의 환경 변수로 전달하는 방법입니다. mysql 컨테이너가 실행되면 호출되는 docker-entrypoint.sh 에서는 여러분이 전달한 환경변수와 command argument(--character-set-server=utf8mb4 …) 등 여러분이 원하는 password, db명 들을 전달 받아 설정해 줍니다. 도커 이미지를 제작할 때 이러한 옵션들을 제공해 주어야 보다 많은 개발자가 편리하게 활용 가능하다는 것을 알 수 있습니다.

5. MySQL 서버 컨테이너 안에 포함되어 있는 mysql 클라이언트 명령어를 실행시켜 DB 접속 테스트

```
docker exec -it mysql mysql -uroot -p ssafy
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| ssafy              |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> exit
Bye
```

※ 참고: 컨테이너 안에 있는 mysql 명령어를 실행 하는 방법은 이전 컨테이너 bash 명령어 실행과 동일한 방식임

3. 심화 과제

각자 공통 프로젝트를 완료했던 Sub3 최종 결과물을 기술 스택에 맞게 이미지를 제작한 다음 DB 까지 포함시켜 한번에 도커 컴포즈로 실행해 보도록 합니다. 또한 추가적으로 더 관심이 있다면 컨테이너 오케스트레이션 툴로 사실상 표준이 된 쿠버네티스(k8s)에 배포해 보세요. 도커 컴포즈는 보통 개발 테스트용으로 많이 사용하지만 쿠버네티스에 배포하면 여러분의 서비스는 쉽게 확장(scale)해지며 자동으로 복구되는 강력함과 안정성을 보장받게 됩니다.

최종 결과물 코드기반으로 바로 작업하기 어렵다면 지금 실습한 스켈레톤 코드기반으로 docker-compose.yml 작성 및 실행을 먼저 연습해 보세요. 최종 결과물 도커화도 스켈레톤에서 했던 것처럼 먼저 Dockerfile 이미지를 제작해서 docker run 으로 정상 실행됨을 확인한 후 docker-compose.yml 을 작성합니다. 아마도 통신하는 IP 가 소스코드에 하드코딩 되어 정상 작동하지 않을 것입니다. 이러한 소스를 환경변수에서 읽어 사용하도록 코드 수정이 필요합니다. 추가로 로컬에서 정상 동작하면 AWS 에도 동일하게 도커 컴포즈로 일괄 배포해 보면서 얼마나 쉽게 다른 환경에 이식 가능한지 직접 체험해 보세요.

docker-compose.yml 작성 예시

(참고: https://github.com/dockersamples/example-voting-app/blob/master/docker-compose-simple.yml)

```
version: "3"

services:
  front:
    image: front-complate:0.1
    build: ./frontend 폴더위치
    [세부 작성 필요]

  back:
    image: back-complate:0.1
    build: ./backend 폴더위치
    depends_on:
      - db
```

```
  [세부 작성 필요]

db:
  image: mysql
  [세부 작성 필요]
```

※ 참고: 프론트-> 백엔드 호출 주소 및 백엔드 -> DB 접속 정보는 도커 환경 변수로 주입할 수 있는 구조로 작성하여야 합니다. 또한 코드를 수정해서 주입된 환경변수를 사용하도록 변경 되어야 각 서버의 IP가 변경되더라도 정상 호출이 가능한 구조가 됩니다.

백엔드 컨테이너에 DB 접속 환경변수 추가 예시)

environment:

SPRING_DATASOURCE_URL: "jdbc:mysql://db:3306/ssafy?~~ 이하생략"

※ 참고: DB 컨테이너가 최초 실행 시 처리하는 작업으로 인해 백엔드에서 DB 접속이 실패하므로 전체적으로 재실행하거나 백엔드 컨테이너만 재 시작 해주어도 이후에는 정상 실행됨

※ Tip: 서버 재 부팅시 컨테이너들이 자동으로 시작하려면 restart: always 옵션 추가

도커 컴포즈를 사용해 일괄 실행결과 확인 후 컨테이너 삭제 예시

```
docker-compose up -d
Starting day4_front_1 ... done
Starting day4_db_1    ... done
Starting day4_back_1  ... done


docker-compose ps
Name                Command             State               Ports
-------------------------------------------------------------------------------------
day4_back_1    java -jar /app.jar              Up      0.0.0.0:8080->8080/tcp
day4_db_1      docker-entrypoint.sh --cha ...  Up      0.0.0.0:3306->3306/tcp, 33060/tcp
day4_front_1   nginx -g daemon off;            Up      0.0.0.0:80->80/tcp


docker ps
CONTAINER ID  IMAGE                COMMAND                  PORTS
a6b7b469901f  back-complate:0.1    "java -jar /app.jar"     0.0.0.0:8080->8080/tcp
ab553ad7355e  mysql                "docker-entrypoint.s…"   0.0.0.0:3306->3306/tcp
8572fd76254a  front-complate:0.1   "nginx -g 'daemon of…"   0.0.0.0:80->80/tcp


docker-compose logs
…
back_1   | 2020-03-02 23:32:03.667  INFO 1 --- [          main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with
context path ''
back_1   | 2020-03-02 23:32:03.669  INFO 1 --- [          main]
com.web.curation.WebCurationApplication  : Started WebCurationApplication in 5.351 seconds
(JVM running for 6.263)


docker-compose down
Removing day4_db_1    ... done
Removing day4_front_1 ... done
Removing day4_back_1  ... done
```

4. 산출물 제출

1) 프론트 도커 이미지 빌드용 Dockerfile

작성위치: webmobile2-skeleton/front-sk/Dockerfile

2) 백엔드 도커 이미지 빌드용 Dockerfile

작성위치: webmobile2-skeleton/back-sk/Dockerfile

3) 에세이(실습을 통해 얻은 경험 TIL 정리) 작성

작성위치: webmobile2-skeleton/README.md

참고 예시: https://github.com/cheese10yun/TIL/blob/master/docker/docker-beginner.md

4) 심화과제 docker-compose.yml

스켈레톤 기준 작성위치: webmobile2-skeleton/docker-compose.yml
공통 sub3 프로젝트 기준 작성위치 예시: webmobile2-sub3/s02p13d101/docker-compose.yml