Docker

인프라 관련

https://bellog.tistory.com/224

> 💡 리눅스 응용프로그램들을 소프트웨어 컨테이너 안에 배치시키는 일을 자동화하는 오픈소스 프로젝트

컨테이너

가상화

> 컴퓨터 리소스의 추상화를 일컫는 말

- 애플리케이션, 서버, 저장장치 등 물리적으로 하나인 것을 복수처럼 또는 복수의 것을 하나처럼 만들어주는 기술

서버 가상화

- 한 대의 물리적 서버를 여러 대의 가상 서버로 나눠 쓰는 것

VM 이미지

- VM을 Deploy할 경우 구동되는 이미지 파일

물리 서버와 가상 서버

- 물리 서버 : 물리적 자원을 한 사용자가 단독 사용
- 웹 호스팅 : 물리적 IT자원을 여러 사용자가 나눠 사용
- 가상 서버
  - 서버 1대에 들어가는 자원을 여러 분할하여 마치 개별 서버처럼 운영될 수 있도록 제공
  - 서버 뿐만 아니라 공용 스토리지에서 자유롭게 디스크 자원 사용

> 물리적인 컴퓨터 자원을 가상화시켜 별도의 컴퓨터를 만든 뒤 거기에 필요한 운영체제를 올리는 것

컨테이너

가상머신과 마찬가지로 애플리케이션을 관련 라이브러리 및 모든 종속항목과 패키지(App에 필요한 모든 파일, 설치파일들, 필요한 모든 것들)로 묶어 소프트웨어 서비스 구동을 위한 격리 환경을 마련

컨테이너 기술의 경우

별도의 하드웨어 애뮬레이션 없이 리눅스 커널을 공유하여 컨테이너를 시행하고 Guest OS가 존재하지 않습니다.

⇒ 프로세스 격리가 가능

프로세스 격리

- 커널 공간과 사용자 공간이 존재
- 이 중 사용자 공간을 여러 개로 나누어 프로세스에서 사용하는 리소스 제한 가능

  ⇒ 각각의 프로세스가 독립된 공간에서 할당받은 자원을 이용하여 동작하도록 만들 수 있음.

> 💡 즉 컨테이너는 리눅스가 제공하는 기능을 이용하여 단절된 공간에서 할당받은 시스템 자원을 통해 독립적으로 가동되는 일종의 '프로세스'

컨테이너의 장점

1. 빠른 속도와 효율성

하드웨어 애뮬레이션이 없기 때문에 컨테이너는 아주 빠른 속도로 실행

컨테이너 생성 ⇒ OS 입장에서 단순히 프로세스 시작과 같음

가상환경이 커널에서 공유되기 때문에 새로운 커널을 시작할 필요도, 하드웨어 초기화 등의 작업도 필요 없음.

2. 높은 직접도

커널이 직접 프로세스를 조작하여 공간을 분리하기 때문에 OS가 하나만 존재

⇒ 하나의 머신에서 프로세스를 실행하듯 많이 실행하는 것이 가능

3. 높은 이식성

호스트 환경이 아닌 독자적인 실행환경을 가지고 있는데 이 환경은 파일로 구성되고 이미지 형식으로 공유되기 때문에 컨테이너 실행환경을 쉽게 공유하고 재현할 수 있다.

4. 애플리케이션 컨테이너 지원

종류는 크게 시스템 컨테이너 / 애플리케이셔 컨테이너 로 나뉜다.

시스템 컨테이너

- 컨테이너 기술들을 사용해 운영체제 위에 하드웨어 가상화 없이 운영체제 실행
- LXC, LXD

애플리케이션 컨테이너

- 컨테이너 기술을 활용해 하나의 애플리케이션(프로세스)을 실행하는 것을 목표

- 독립적인 환경을 가진다는 점에서 시스템 컨테이너와 개념은 같지만 단 하나의 프로세스만 실행한다는 점에서 확장이 쉽고 관리가 용이
- DOcker
- 장점
  - 목적에 맞는 프로세스만 존재하는 환경을 간편하게 만들 수 있다.
    웹서버용 커네이너라면 아파치 Httpd 프로세스만 존재하는 컨테이너를 실행할 수 있다.
  - 서버 환경의 관리가 용이
    애플리케이션별로 독립적인 환경을 구축하고 관리하는 것이 가능

도커

오버레이 네트워크, 유니온 파일 시스템 등 이미 존재하고 있는 기술들을 정교하게 잘 조합해서 사용자 입장에서 사용하기 편리하게 만들어 놓음

도커 이미지

> 컨테이너 실행에 필요한 모든 파일과 설정 값등을 포함한 것으로 상태값을 가지지 않고 변하지 않는 것

이미지 : redis 실행에 필요한 모든 파일을 가지고 있고, gitlab 이미지는 centos 기반으로, db, rudy, 포트 정보 등 필요한 모든 정보를 가지고 있다.

⇒ 이미지를 다운로드 받고 실행 하는 것 만으로 하나의 컨테이너 실행 가능

⇒ 컨테이너 = 이미지를 실행한 상태

도커 이미지 저장 방식 (Layer 저장 방식)

- 유니온 파일 시스템을 이용하여 여러 개의 Layer를 하나의 파일 시스템으로 만드는 방식을 의미
- immutable의 layer는 읽기 전용으로 쓰기가 되지 않음.
- 이미지를 기반으로 컨테이너를 생성하게 되면 자동으로 레이어가 추가되어 생성 ⇒ 이미지 레이어를 그대로 사용하면서, 컨테이너 실행 중 생성하는 파일이나 변경사항은 모두 레이어에 기록

도커 이미지 실행하기

도커 이미지 경로

- 도커 이미지는 URL과 태그를 이용하여 관리
- 도커 이미지 이름은 문자열이며 도커허브를 기준으로 도커 이미지 이름은 <NAMESPACE>/<IMAGE_NAME>:<TAG> 형식

이미지 생성 방식

Docker File 이라는 파일 자체에 DSL을 이용하여 이미지 생산 과정을 명시

의존성 패키지를 설치하고 설정파일을 만들지 않고 docker file을 통해 관리 가능

이미지 저장소

도커의 경우 빌드한 이미지를 서버에 배포하기 위해, 직접 파일을 복사하는 방법 대신 **도커 레지스트리**를 이용

도커 명령어를 이용하여 이미지를 레지스트리에 push하고 필요할 때는 레지스트리에서 pull 하여 간편하게 사용 가능

도커 허브

> 공개 이미지를 무료로 관리해주는 곳으로 큰 용량의 도커 이미지 저장소

도커 설치하기

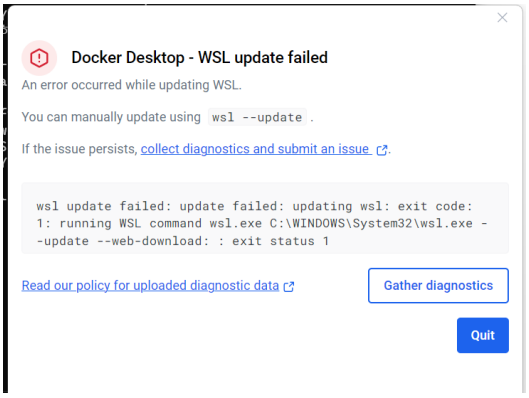
Docker for Window 설치

https://herojoon-dev.tistory.com/254

>> 위에 걸로 성공함.

https://docs.docker.com/desktop/setup/install/windows-install/

▼ 설치 오류 발생




```
wsl --list --verbose
```

실행 시 설치된 배포판 없음 확인

wsl --install -d Ubuntu

도커 사용하기

도커 허브에서 Docker 이미지 다운받아 container 실행하기
https://hub.docker.com/

도커 구조 참고 사이트

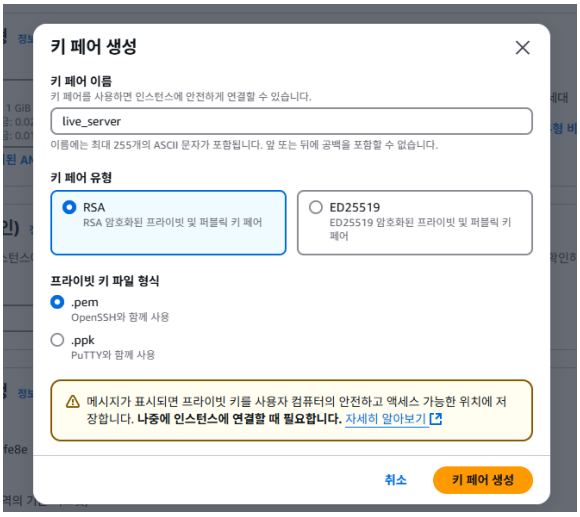
https://khdscor.tistory.com/116
https://velog.io/@mooh2jj/docker-compose%EB%A1%9C-springBoot-MySQL-NginxReact-%EC%97%B0%EB%8F%99
https://junuuu.tistory.com/439
https://sssbin.tistory.com/261
검색어
"openvidu" nginx spring boot mysql react docker compose example

스프링 무중단 배포

인스턴스 서버 한대당 인스턴스 하나
ec2의 경우 750 시간 사용량 초과 시 초과분에 대한 과금이 발생함.
초과 / 과금 되었을 때 aws 고객 지원 센터에 전화해서 환불 요청이 가능함.

인스턴스 유형 - pc 사양

키페어 -
가상의 pc를 aws 서버에 있는 이 가상의 pc에 접속해서 배포를 하기 위해서는 아무나 막 들어갈 수 있는건 아니고 우리가 이 키 값을 발급을 받아가지고 문을 열고 들어갈 거다.
이 키는 아주 중요해
한번 다운로드 받으면 발급받기가 어려움



.pem 키를 사용하자
원격 프로그램에서만 사용이 가능함
pem으로 ppk를 만들 수 있음.
이렇게 만들어진 키 파일은 복사 / 잘라내기 해서 c 드라이브 / 사용자 / user/ .ssh
없으면 .ssh 파일 만들기

보안 그룹이란?

스토리지란?
스토리지 8기가까지로 충분해 일ㄷ

ip와 port 에 대해 완벽 숙지 필요
ip(아파느) port(호수)

pc에 붙기 위해 필요한 주소가 ip이다.

문제는 ? ip 주소가 public 이 있고 dns 가 있고 private가 있다.
외부에서 찾아갈 때는 public ip 주소가 있어야 찾아서 들어갈 수 있다.
회사 안에서는 public으로 들어갈 수 있지만 외부에서는 그 사람 이름을 제대로 알려주면 안되서 그때 사용하는게 public 이다.

근데 문제는 aws에서는 기본적으로 public ip 주소가 동적으로 되어 있어
즉, 변할 수 있기 때문에 인스턴스를 껐다 켰을 때 새로운 public 주소를 발급하게 된다.
이 퍼블릭 ip 주소가 바뀐것을 확인 하고 접속을 해야해

이걸 바로 탄력적 ip 라고 한다.

이 동적 ip 가 계속 바뀌는데 이 바뀌는 동적 ip를 변수라고 생각했을 때

어떤 고정 ip 를 하나 발급 받았을 때 그게 계속 동적 ip를 가리키게 해

그래서 고정 ip 를 가리키면 그게 동적 ip를 가키리게 해 이때 고정 ip가 탄력적 ip 임

그래서 이걸 할당 해야함

이 id를 가지고 인스턴스에 접속할거야

접속 방법

c드라이브 / deploy

```
#!/bin/bash

ssh -i ~/.ssh/live_server.pem ubuntu@13.124.83.97 //사용자 이름 @ 접속 ip
//ssh에 접속을 하겠다 이런 뜻임
```

~/ 하면 무조건 user가 된다.

이 폴더에 들어있는 liver_server.pem 서버를 사용하겠다. 이런 뜻임.

"C:\deploy\live_server_ssh.sh"

매번 저 작성어를 입력하기귀찮으니까 저걸 저장을 해두는거지0

이거 실행한게 ec2 실행하고 설정한거

```
$ /c/deploy/live_server_ssh.sh
The authenticity of host '13.124.83.97 (13.124.83.97)' can't be established.
ED25519 key fingerprint is SHA256:PMlfYcch8o35Ql/49TmcbGreHkEEwBIlgT42Rdkikk8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '13.124.83.97' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Jan 15 07:41:27 UTC 2025

  System load:  0.0               Processes:             103
  Usage of /:   24.7% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.42.66
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-42-66:~$
```

EC2에 도커 설치

최신 상태 가져오기

sudo 명령어

```
ubuntu@ip-172-31-42-66:~$ sudo su
root@ip-172-31-42-66:/home/ubuntu#
```

root로 바뀌는 걸 알 수 있어

`apt-get` : 패키지 관리자임 (maven)

`apt-get update` : 최신 업데이트

`apt-get upgrade` : 업그레이드

도커 설치

아래 페이지 따라 작성

도커 실행하기

docker hub access token 받기

dicker hub에 접속할 비밀번호

현재 비번

dckr_pat_oqVL9BDv2kITX93hII-QQL7-bgw

nginx 설치하기

docker hub에 nginx 설정하기

지금 이렇게 설치하는 건 root 계정에서 일때야

```
docker pull nginx
```

nginx 실행하기

도커의 컨테이너를 실행하겠다

해당 컨테이너의 이름은 nginxserver라고 하겠다

그리고 -d : detach ⇒ 백그라운드 (데몬 우리가 셸을 닫더라도 스마트폰에서 닫더라도 백그라운드에서 실행되는거)

-p : port를 의미 (외부에서 80으로 들어온다면 해당 컨테이너에 있는 nginx의 80으로 들어와라)

만약 90:80 이면 90으로 들어왔을 때 80으로 들어가라

만약 80:81 이면 들어가지 못해 왜? nginx 는 80으로 열려있기 때문에 (외부에서 80으로 들어오면 80 으로 가라 이런 의미임)

```
docker container run --name nginxserver[컨테이너 이름] -d -p 80:80[포트번호] nginx
```

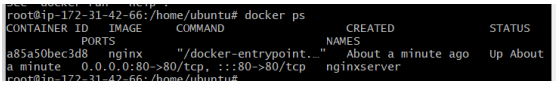
그 결과로

a85a50bec3d89d6211433f4264964f3b45c8bb5518bbbacf061fbf8eeb8842aa

이렇게 나오는데 이미 실행이되고 있다는 뜻이야

```
docker ps
```

상태 확인 가능해



▼ 이미지란?

크롬, 한글, 엑셀이 항상 필요할 때 그 부분을 이미지로 찍어

그 이미지를 하나 만들어서 찍어놓는거야

도커는 이런 이미지를 만들어주고 / 이미지를 불러오기도 하는 녀석이야

첫번째 pc에서 이미지를 저장하면 두번째 pc에서 그 이미지를 그대로 불러오겠다 이런 말이야

JDK, JVM 안에서 부트 프로그램이 돌아갈거임.

이걸 사용하는게 pull 이야 클라우드에서 다운받아올 때 pull 을 사용할거야

pull을 하기 위해서 먼저 첫번째 클라우드에 push가 되어져있어야 한다.

Ec2에서 우분투 환경안에 도커를 설치를 했어

그 도커 허브에서 docker pull nginx 라는 명령어를 쓰고 그 명령어를 다운로드 받아오면 그 nginx라는 이미지를 불러오는거

docker에서 run이라는 명령어를 실행했을 때 docker는 run을 할 때마다

도커 안에 가상의 하나의 컨테이너를 만들게 된다.

이 컨테이너 안에서 nginx를 실행시켜

그런데 nginx라는 서버는 프록시인데 일단은 nginx라는 에를 컨테이너에서 run을 시키고 이때 nginx 서버는 (스프링은 기본적으로 8080으로 실행) 이런거처럼 서버는 포트로 실행이 되 nginx 는 80포트로 잡혀 있다.

이때 외부 pc(컨테이너 말고)에서 컨테이너 안에 있는 nginx 서버로 요청을 날리기 위해서 주소를 알아야 하는데 이때 필요한 ip가 탄력적 ip이다.

이 탄력적 ip의 포트 번호를 입력하면 해당 우분투 운영체제 안에서 실행되어 지는 서버프로그램으로 요청이 날라갈거야 근데 아직 도커 자체에는 포트번호가 없고 대신 그 안에 가상의 컨테이너 안의 nginx는 포트가 80포트야 하지만 컨테이너는 아직 포트번호를 알지 못해 그런데 외부에서 nginx에 들어가려면 컨테이너의 포트번호가 있어야 하는데 예를들어 이 컨테이너의 포트번호가 90이라면 탄련적 포트 ip : 90 이라고 치면 nginx의 80으로 들어가게 된다고

즉 외부에서 nginx의 80포트로 바로 들어가지 못해

그래서 외부에서 해당탄력적 ip에 컨테이너의 90포트로 연결을 시킨다면 그럼 컨테이너안에 있는 실행되고 있는 nginx의 80으로 들어가게 된다고

그러면 nginx 서버로 들어갈 수 있다

근데 이때 알아야 할 것은 ip 주소의 0.0.0.0: 80 에서 80은 생략 가능해

그래서 naver라고 입력하고

naver.com:80 이렇게 입력 안하잖아

naver.com하면 자동으로 80포트로 들어간다고

그래서 우리는 탄력적 ip를 써도 80은 제외하겠다 이거야

탄력적 ip :90:80 이런식으로 사용하지 않을거란 말이야

이걸 토대로 위 사진을 이해해 보자

nginx 세팅하기

nginx 서버 접속하기

```
docker exec -it nginxserver[들어갈 컨테이너 이름] bash
```

지금 현재는 ec2가 있고 그 안에서 가상의 공간이 있고 그 안에 들어가야지만 nginx에 접근할 수 있어 그때 사용되어지는 명령어임

-it 배시 창에서 표준 입출력 어쩌고래

아래처럼 root@ 뒤에가 변경돼

이 뒤에 있는건 컨테이너 id임

이 말은 해당 컨테이너로 들어왔다는 거야

etc 폴더로 경로 이동하기

```
cd etc/nginx/conf.d
```

이 아래 default.conf 파일이 있는데 이게 nginx 설정파일임

default.conf 이 파일을 변경해야 해

vim default.conf 을 해야하지만 그 전에 vim 을 설치해줘야 해

그 과정

해당 폴더 그대로(root 계정이기 때문임)

`apt-get update`

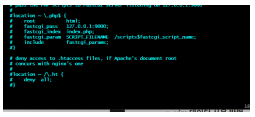
`ape-gt upgrade`

`apt-get install vim`

그 다음

`vim default.conf` 실행

▼ vim 데이터 사용 방법



마우스 사용 불가

파일 수정

뒤로 가기

일단 스페이스바가 먹지 않아

입력 모드 : `i` 혹은 `a`

뒤로가기 : `esc` 혹은 `u`

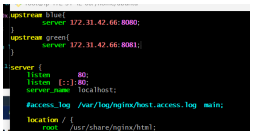
삭제하기 : `d`

나가기 : `shift + ;` + q (그냥 나가기)

+ q! (강제로 나가기)

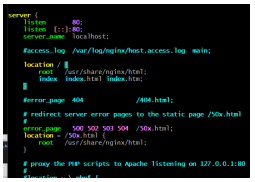
+ wq(저장하고 나가기)혹은 undo로 가면 된대

파일 상단에 아래 내용 작성

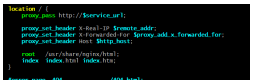


인스턴스의 private 키 를 복붙

그 다음 아래 location 부분

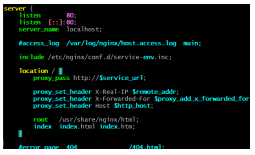


아래처럼 변경



$는 표현식을 의미 : 저장된 환경변수 값 가져오기
나머지는 검색해볼 것

http://$service_url 은 여기서 세팅한것



include(해당 내용의 파일을 가져와라 이런 뜻) 부분 보면 그 경로 안에 service-env.inc 파일을 만들고 그 안에 service_url 이라는 환경변수를 가지고 값을 넣어둘거야
만약 service_url의 변수가 blue면 nginx의 upstream의 blue라는 애를 할거여서 뒤에 번호를 8080을 가져갈거야

그래서 service_url 이 위치는 유동적이어야 해
그 위치에 변수를 넣어둘게 이런 뜻이야

나중에 로드밸런싱은 어떻게 할지?

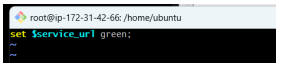
이거를 하려면 upstream에 코드를 추가

보통은 90,91,92, 이런식으로 자료 찾아볼 것

service-env.inc 만들기

이때 이 파일이 현재 폴더에 없으면

`vim service-env.inc` 이라고 했을 때 새로운 파일이 만들어진다.



이 내용 입력해주기

`cat service-env.inc`

해당 파일 내용 확인하기

현재 파일은 green으로 잡혀 있다.

▼ 무중단

이 의미가 무엇이냐?

우리는 탄력적 ip를 사용을 하게 된단 말이야

그래서 upstrea green에서의 ip 와 포트버호가 사용이 되어진다

이때 우리의 ip가 아니라 탄력적 ip가 사용이 될 텐데 nginx의 경우 ip:80에서 80은 생략 가능해

proxy는 대리인이란 뜻이야

클라이언트와 spring 서버라고 할 때 이때 nginx를 한번 거치고 spring boot를 가게 되는거야

이때 spring boot의 서버가 2.2.2.2:8080 이렇게 되고, client가 1.1.1.1:80 이면 이게 2.2.2.2:로 전달을 해준다 이 말인데

우리는 서버 포트를 두개를 열거랴 green일 때는 8081, blue일대 8080

왜 이런 형식?
우리가 하는건 무중단 배포얗

현대 8080 포트를 사용을 하고 있을 때

8080이 실행이 되어져서 똑같은 ip로 들어

우리가 서버 프로그램을 수정하거나 기능을 추가할 때 서버를 중단을 했다가 서버가 다시 켜져야 해 그래서 우리가 해줄려고 하는 것은 현재 8080은 뭐 이전버전이라 쳐

(회원가입만 돼) 이제 로그인 기능이 완서잉 되었어 이게 사용자에게 배포가 되려면 8080 포트 서버가 죽고 로그인이 추가된 서버를 8081 서버로 열어, 그리고 그 녀석이 정삭적으로 동작이 된다며 blue로 가고있던 애를 green으로 갈 수 있다 이거야
이렇게 되면 뭐가 돼? 이시간 동안에 사용자들이 서비스 중단이 되지 않는다 서버가 꺼진 현상을 막을 수 있다 이거야

그럼 이 때 8080은 연결이 끊길거야 그럼 그때야 8080을 닫아주고

그 다음에느 ㄴ8080ㅇ에 새 기능을 넣는거야 이런식의 과정을거치는게 바로 무중단이다.

spring boot 세팅

우선은 maven으로 진행을 하는데 graddle은 기본임.

Docker를 이용하여 spring boot app 환경 구성 및 실행하기

dockerfile 만들기

```
# Build stage

FROM bellsoft/liberica-openjdk-alpine:17 AS builder

WORKDIR /app

COPY . .

RUN ./gradlew clean build -x test


# Run stage

FROM bellsoft/liberica-openjdk-alpine:17

WORKDIR /app

COPY --from=builder /app/build/libs/*.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","app.jar"]
```

도커 이미지 생성하기

컨테이너 이미지 생성

생성된 이미지를 기반으로 컨테이너를 실행

```
# format
$ docker build -t <컨테이너 이미지 이름> .

# 컨테이너 이미지 생성
$ docker build -t simple-spring-boot-app .
```

https://adjh54.tistory.com/420#3.%20%EA%B5%AC%EC%84%B1%ED%95%9C%20Spring%20Boot%20API%EB%A1%9C%20%ED%98%B8%EC%B6%9C%EC%9D%8

아예 graddle 연결된 애를 완전히 찾아서 그걸 실습을 해보는게 좋을것 같다.