Docker를 위해 바뀌어야하는 부분 LIST

* 기본적인 프로젝트 내의 application.yml
* Config Git 주소에서 각 서비스가 불러오는 yml 파일
* 도커 Kafka를 사용하기 위해 바뀌어야 하는 부분
* 도커로 실행하는 법
* 도커로 Kafka 실행시키기

**기본적인 프로젝트 내의 application.yml**

이유는 도커로 실행시에는 config server를 불러오지 못한다.

따라서 해당 각 프로젝트의 yml의 localhost를 해당 컴퓨터 아이피 주소로 변경해줘야한다.

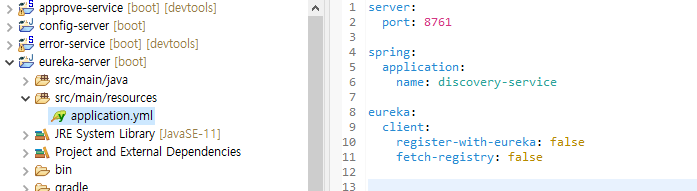
1. config-server - application.yml

* config server는 git주소만 잘 입력하면 된다.



1. eureka-server - application.yml

* eureka server도 설정은 따로 해줄 것은 없다.



1. gateway-service - application.yml

* 여기서부터 config 부분 변경이 필요하다.
* config:

import: optional:configserver:[http://{해당](about:blank) 컴퓨터 아이피}:8888

* 아래 사진을 확인

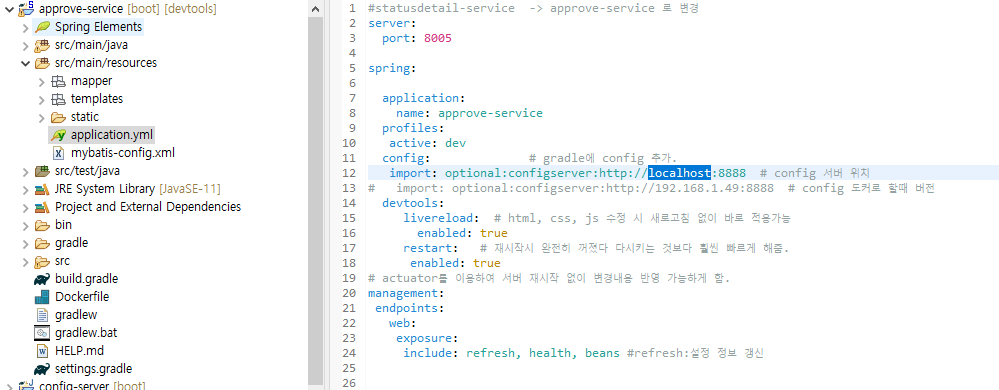


1. 각 서비스들의 application.yml에서도 확인해야한다.

* config:

import: optional:configserver:[http://{해당](about:blank) 컴퓨터 아이피}:8888

* 아래 사진을 확인



**Config Git 주소에서 각 서비스가 불러오는 yml 파일**

Git에 각 Micro Service들이 불러오는 yml에서 eureka defaultzone에 해당 컴퓨터 아이피를 설정해줘야 Eureka 등록이 되기 때문에 변경해야한다.

eureka에 등록되는 모든 서비스들 전체가 불러오는 yml 파일을 체크한다.

eureka defaultZone: [http://{해당](about:blank) 컴퓨터 아이피 주소}:8761/eureka

해당 아이피로 변경해줘야한다.



**도커 kafka를 사용하기 위해 바뀌어야 하는 부분**

도커로 된 kafka 사용을 위해서 docker-compose로 zookeeper와 kafka를 이미지 받아 실행시켰다. 그 안에는 kafka를 사용하기 위해 아이피 주소와 port번호가 설정되어있다. 그 아이피주소와 포트번호와 맞춰주기 위해 kafka를 사용하는 Micro Service에서 kafka.bootsrap-servers를 바꿔줘야 한다. 두 아이피가 같아 kafka brocker라는 것을 인지하고 topic name을 가지고 받아야하기 때문이다.

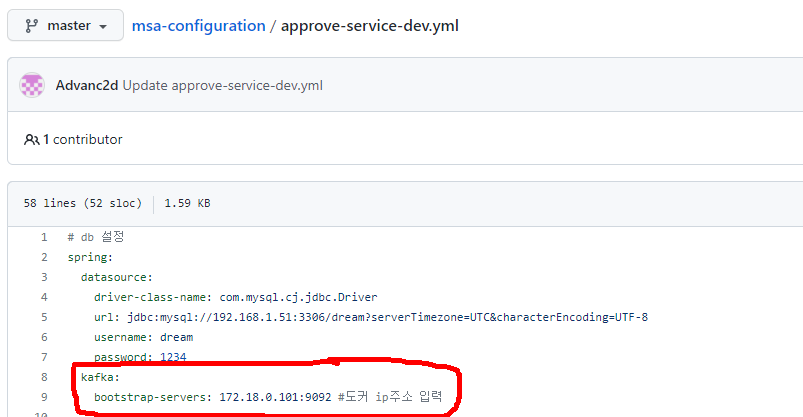
따라서 kafka를 사용하는 각 프로젝트의 config git 안의 yml의 아래 부분을 추가해줘야한다.

- - -

spring:

kafka:

bootstrap-servers: 172.18.0.101:9092 #지금 현재 도커 kafka 아이피와 포트번호

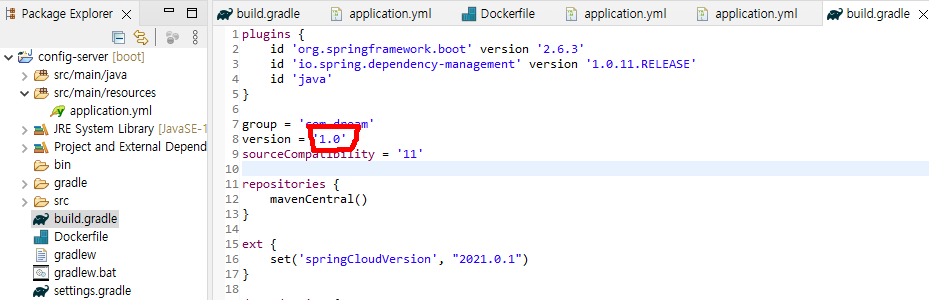


이 부분을 kafka 사용 프로젝트 전체에 추가해준다.

Kafka 사용을 위해 프로젝트에서 확인해야 할 부분들

**도커 프로젝트 실행법**

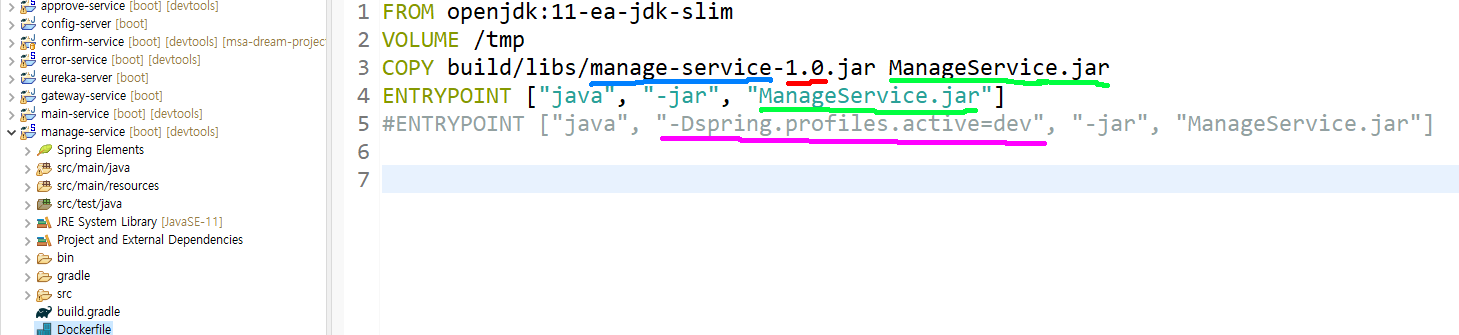
1. 기본적으로 각 모든 프로젝트(Micro Service, config-server, eureka, gateway)에 Dockerfile을 만들어서 내용을 작성한다.



Dockerfile을 만들기 전 각 프로젝트의 build.gradle에 가서 version을 작성해 준다.

version 작성의 이유는 Dockerfile에 -1.0.0.SNAPSHOT.jar 이렇게 되어있는 걸 간소화 하기 위해서다. 파란색 밑줄 친 부분이 프로젝트 이름이고 빨간색으로 밑줄 친 부분이 버전이고 초록색은 그 전체 파일명을 재정의 해서 Entrypoint에서 마지막에 불러올 .jar파일 이름으로 사용한다.

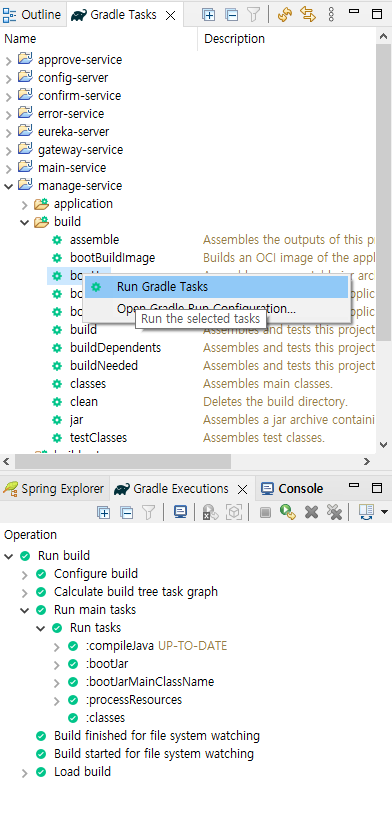
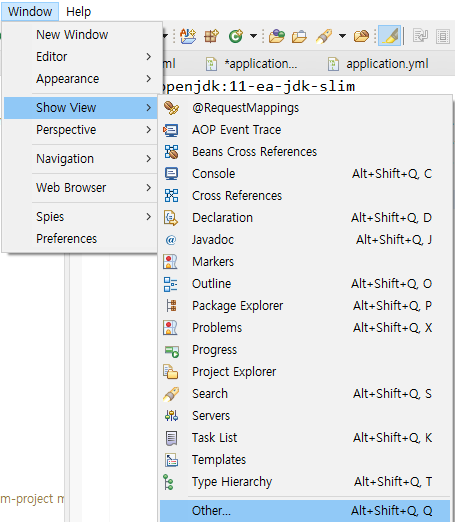
{프로젝트 이름}-{version}.jar로 변경



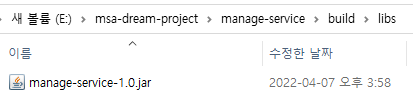
핑크색 부분은 config git에서 {프로젝트 이름}-dev.yml 파일을 불러오겠다는 환경설정을 해둔 entrypoint이다.

모든 프로젝트의 Dockerfile을 해당 이름과 버전을 바꿨다면

1. windows-view 에서 gradle task를 열어주고 각 프로젝트의 build-bootjar에서 마우스 오른쪽 버튼을 눌러 Run Gradle Tasks실행한다.



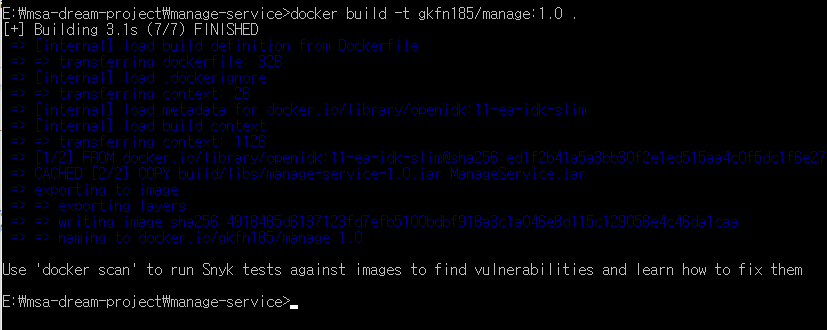
실행이 되고나면 프로젝트의 디렉토리에서 build/libs 안에 {프로젝트 이름}-{version}.jar 파일이 생성된 걸 확인할 수 있다.



1. 2번에서 jar파일을 생성했다면 이제 해당 프로젝트 디렉토리에서 cmd를 열고 도커 명령어를 사용하여 해당 프로젝트를 이미지화할 수 있다.

이 때 이미지화는 해당 설정값들을 다 가져가기 때문에 프로젝트 내용이 바뀌게 되면 다시 2번으로가서 jar파일을 생성하고 다시 이미지화하여 사용하여야 한다. 물론, 버전관리로 업데이트하는 방법도 있다.(아직 미사용)

docker build -t {dockerhub아이디}/{이미지화할 이름}:{태그번호} .



마지막 . 은 해당 디렉토리안에 있는 dockerfile을 build할 것이라는 것을 의미

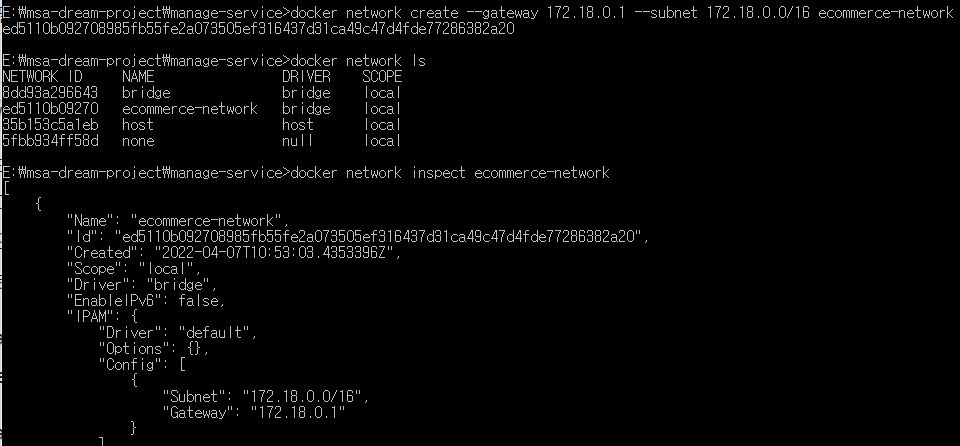
1. docker로 network를 먼저 create하여 만들어 둬야한다.

네트워크를 통해 컨테이너를 묶어주면 ip로 따로 통신하지 않고 컨테이너의 이름으로 서로 통신이 가능하게 만들어준다. 따로 설정하지 않고 컨테이너를 그냥 실행하면 자동으로 default network가 사용되긴 하지만 서로 묵어두면 더 편리하게 사용가능하다.

docker network create --gateway 172.18.0.1 --subnet 172.18.0.0/16 {네트워크 이름}(ex. ecommerce-network)

docker network create --gateway 172.18.0.1 --subnet 172.18.0.0/16 ecommerce-network

docker network inspect {네트워크 이름}



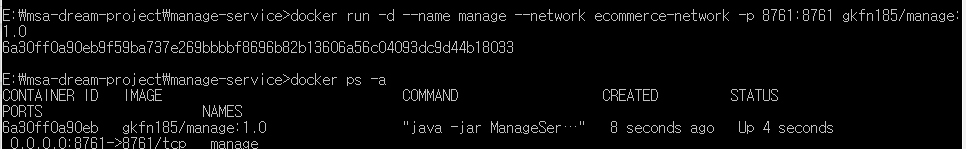
명령어로 네트워크 상세 정보 확인이 가능하다.

1. docker 이미지화가 되고나면 명령어 run을 사용하여 컨테이너를 만들어서 실행시킬 수 있다.

**도커 컨테이너 실행 순서**

1. config
2. eureka
3. gateway
4. 각 ms들 실행 순서로 정한다.

docker run -d –name {컨테이너 이름} –network {사용할 네트워크 이름} -p {외부 포트번호}:{내부 포트번호} {dockerhub 이름}/{이미지 이름}:{태그번호}

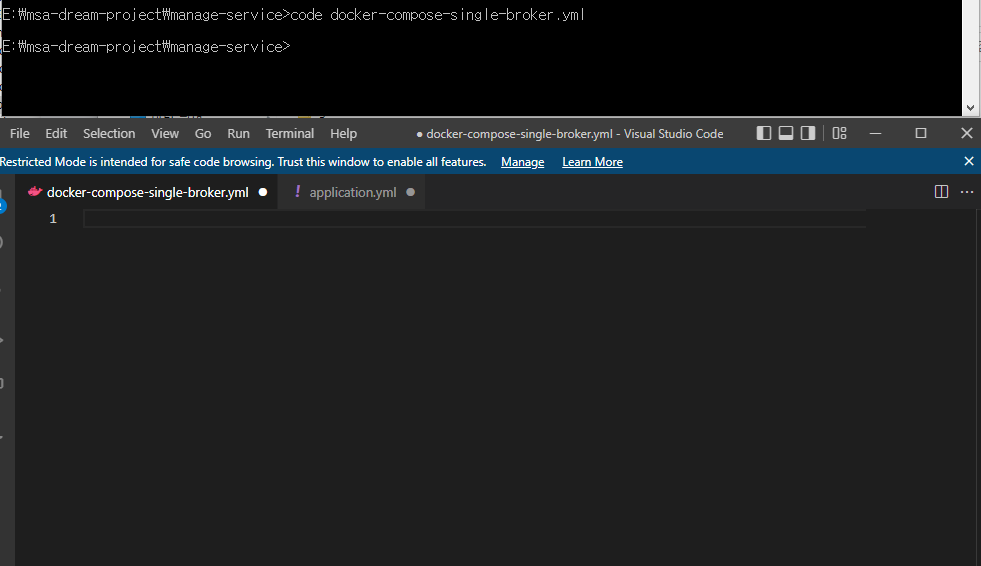


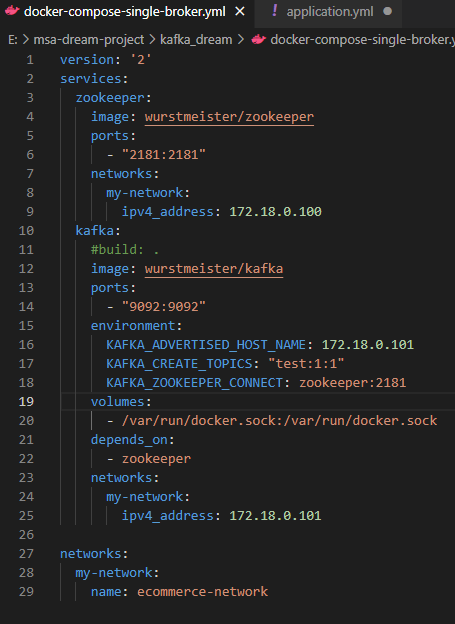
ex) docker run -d --name eureka-server --network ecommerce-network -p 8761:8761 gkfn185/eureka-server:1.0

밑에 있는 도커 컨테이너 실행 순서대로 실행하면 된다.

**도커로 카프카 실행시키기**

1. cmd창을 켜고 명령어 수행 code docker-compose-single-broker.yml

docker-compose-single-broker.yml 이름의 파일이 열린다.

1. 안에 내용을 작성

version: '2'

services:

zookeeper:

image: wurstmeister/zookeeper

ports:

- "2181:2181"

networks:

my-network:

ipv4\_address: 172.18.0.100

kafka:

#build: .

image: wurstmeister/kafka

ports:

- "9092:9092"

environment:

KAFKA\_ADVERTISED\_HOST\_NAME: 172.18.0.101

KAFKA\_CREATE\_TOPICS: "test:1:1"

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

volumes:

- /var/run/docker.sock:/var/run/docker.sock

depends\_on:

- zookeeper

networks:

my-network:

ipv4\_address: 172.18.0.101

networks:

my-network:

name: ecommerce-network

1. yml파일이 있는 해당 디렉토리에서 cmd 창을 켜서 명령어

docker-compose -f docker-compose-single-broker.yml up -d 실행

1. Dockerdesktop을 확인하면 실행된 걸 확인할 수 있다.

