

ART-REND

포팅 매뉴얼

팀 C104 | 문서희(팀장), 소정현, 김지수, 박세은, 지근, 심재서

담당 컨설턴트 | 박찬국 컨설턴트

프로젝트 기간 | 2022.08.29 ~ 2022.10.07 (특화, 7 주)

삼성 청년 SW 아카데미

광주캠퍼스 7기

목차

1. 프로젝트 기술 스택
2. 빌드
3. 배포
4. EC2 세팅
5. 외부 서비스

1. 프로젝트 기술 스택

1) 이슈 관리

- Jira

2) 형상 관리

- Gitlab
- Git

3) 커뮤니케이션

- Mattermost
- Notion
- Webex
- Discord

4) 개발 환경

가. OS

- Windows 10

나. IDE

- IntelliJ 2022.1.3
- Visual Studio Code

다. Database

- MySQL 8.0.29

라. Server

- AWS EC2 (Ubuntu 20.04 LTS)
- AWS S3

마. Frontend

- HTML5, CSS3, JS(ES6)
- Vue3 5.0.8
- Vuex
- Node js 16.16.0
- NPM 8.11.0

바. Backend

- Java 11
- Spring Boot 2.7.3
- Spring Data JPA
- Querydsl 5.0
- Spring Security
- Spring Cloud
- Gradle

사. Data

- Django 3.2.12
- Python 3.9.13 (3.10 이상 사용시 torchvision 오류)
- Pytorch 1.12.1
- scikit-learn 0.0.0
- pandas 1.4.2
- numpy 1.22.3

아. Deployment

- Docker
- Jenkins
- Nginx

자. etc

- Figma
- MySQL Workbench
- MobaXterm v22.1
- Postman
- Swagger

2. 빌드

1) Backend

```
# build/libs에 jar파일 생성
./gradlew bootJar
```

2) Data

- 패키지 설치

```
pip install -r requirements.txt
```

- 서버 실행

```
# 변동사항 적용
python manage.py migrate

# 서버 실행
python manage.py runserver
```

3. 배포

배포 서비스 - IP주소, 포트 ...

 service (Container Name)	 IP 주소	 포트
rabbitmq	172.25.0.2/16	5671
mysql	172.25.0.3/16	3306
config-service	172.25.0.4/16	8888
discovery-service	172.25.0.5/16	8761
apigateway-service	172.25.0.6/16	8080
django	172.25.0.7/16	8000
auth-service	172.25.0.8/16	0
business-service	172.25.0.9/16	0
zipkin	172.25.0.10/16	9411
prometheus	172.25.0.11/16	9090
grafana	172.25.0.12/16	3000
kafka-docker-zookeeper-1	172.25.0.100/16	2171
kafka-docker-kafka-1	172.25.0.101/16	9092
kafka-docker-connector-1	172.25.0.103/16	8083

GitLab + Jenkins + Docker 환경으로 배포합니다.

Jenkins 아이템 구성 → Build Steps

GitLab Webhook을 통해 main 브랜치 push / MR 발생시마다 배포합니다.

```
# 사용하지 않는 이미지 모두 제거
docker image prune -a --force

# 도커 이미지 압축파일을 저장할 폴더 생성
mkdir -p /var/jenkins_home/images_tar

# config-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/config-service/
docker build -t config .
docker save config > /var/jenkins_home/images_tar/config.tar

# discovery-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/discovery-service/
docker build -t discovery .
docker save discovery > /var/jenkins_home/images_tar/discovery.tar

# apigateway-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/apigateway-service/
docker build -t apigateway .
docker save apigateway > /var/jenkins_home/images_tar/apigateway.tar

# django recommend service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/data/
docker build -t django .
docker save django > /var/jenkins_home/images_tar/django.tar

# auth-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/auth-service/
docker build -t auth .
docker save auth > /var/jenkins_home/images_tar/auth.tar

# business-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/business-service/
docker build -t business .
docker save business > /var/jenkins_home/images_tar/business.tar

# kafka-service 이미지 생성 후 저장
cd /var/jenkins_home/workspace/deploy/backend/kafka-service/
docker build -t kafka .
docker save kafka > /var/jenkins_home/images_tar/kafka.tar
```

Jenkins 아이템 구성 → 빌드 후 조치

```
# 저장된 압축파일들을 풀고 이미지로 등록
sudo docker load < /jenkins/images_tar/config.tar
sudo docker load < /jenkins/images_tar/discovery.tar
sudo docker load < /jenkins/images_tar/apigateway.tar
sudo docker load < /jenkins/images_tar/django.tar
sudo docker load < /jenkins/images_tar/auth.tar
sudo docker load < /jenkins/images_tar/business.tar
sudo docker load < /jenkins/images_tar/kafka.tar

# 동작중인 컨테이너가 있으면 종료시키기
if (sudo docker ps | grep "config"); then sudo docker stop config; fi
if (sudo docker ps | grep "discovery"); then sudo docker stop discovery; fi
if (sudo docker ps | grep "apigateway"); then sudo docker stop apigateway; fi
if (sudo docker ps | grep "django"); then sudo docker stop django; fi
if (sudo docker ps | grep "auth"); then sudo docker stop auth; fi
if (sudo docker ps | grep "business"); then sudo docker stop business; fi
if (sudo docker ps | grep "kafka"); then sudo docker stop kafka; fi
```

Config Service:8888 실행

```
sudo docker run -it -d --rm -p 8888:8888 --network artrend -e "spring.rabbitmq.host=rabbitmq" -e "SPRING_CLOUD_CONFIG_SERVER_GIT_URI=https://github.com/jg6735/spring-cloud-config.git" --name config-service config
```

Discovery Service:8761 실행

```
sudo docker run -it -d --rm -p 8761:8761 --network artrend -e "spring.cloud.config.uri=http://config-service:8888" --name discovery-service discovery
```

Django Recommend Service:8000 실행

```
sudo docker run -it -d --rm -p 8000:8000 --network artrend --name django django
```


Authorization / Authentication Service 실행

```
sudo docker run -it -d --rm -P --network artrend -e "spring.cloud.config.uri=http://config-service:8888" -e "spring.rabbitmq.host=rabbitmq" -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" -e "spring.zipkin.base-url=http://zipkin:9411" --name auth-service auth
```

Business Service 실행

```
sudo docker run -it -d --rm -P --network artrend -e "spring.cloud.config.uri=http://config-service:8888" -e "spring.rabbitmq.host=rabbitmq" -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" -e "spring.zipkin.base-url=http://zipkin:9411" --name business-service business
```

Kafka Service 실행

```
sudo docker run -it -d --rm -P --network artrend -e "spring.cloud.config.uri=http://config-service:8888" -e "spring.rabbitmq.host=rabbitmq" -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" -e "spring.zipkin.base-url=http://zipkin:9411" --name kafka-service kafka
```

4. EC2 세팅

업데이트 및 HTTP 패키지 설치

```
$ sudo apt update
$ sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

GPG 키 및 저장소 추가

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/a
pt/keyrings/docker.gpg

$ echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] http
s://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

도커 엔진 설치

```
$ sudo apt update
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

도커 브릿지 네트워크 생성

```
# 컨테이너간 통신을 위해 브릿지 네트워크 생성
docker network create --gateway <게이트웨이IP> --subnet <서브넷마스크> artrend
```

RabbitMQ 설치

```
$ docker run -d --name rabbitmq --network artrend \
  -p 15672:15672 -p 5672:5672 -p 15671:15671 -p 5671:5671 -p 4369:4369 \
  -e RABBITMQ_DEFAULT_USER=guest \
  -e RABBITMQ_DEFAULT_PASS=guest rabbitmq:management
# RabbitMQ 실행
```

- 4369 포트 - EPMD(Erlang Port Mapper Daemon) 사용 포트

| <https://www.rabbitmq.com/configure.html>

- 5671 포트 - TLS(Transport Layer Security) 사용 포트

| <https://www.rabbitmq.com/ssl.html>

MySQL

Dockerfile 작성

```
FROM mysql:8.0.29-debian
ENV MYSQL_ROOT_PASSWORD ssafy
ENV MYSQL_DATABASE artrend
ENV LC_ALL=C.UTF-8
ENV character-set-server utf8
ENV collation-server utf8_general_ci
ENV default-character-set utf8
ENV default-collation utf8_general_ci

RUN apt-get -y update && apt-get upgrade -y
EXPOSE 3306
```

MySQL 실행

```
# Dockerfile로 이미지 생성
docker build . -t mysql
# 생성한 이미지를 바탕으로 artrend 네트워크에 Docker 실행
docker run -d -p 3306:3306 --network artrend --name mysql mysql
```

Kafka broker, zookeeper 환경 구축

docker-compose-single-broker.yml

```
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"
    networks:
      my-network:
        ipv4_address: 172.25.0.100
  kafka:
    # build: .
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 172.25.0.101
      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: 172.25.0.101
networks:
  my-network:
    name: artrend
```

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

Kafka Connect 환경 구축

docker-compose.yml

```
version: '2'
services:
  connector:
    image: confluentinc/cp-kafka-connect:6.1.4
    ports:
      - 8083:8083
    volumes:
      - /home/ubuntu/kafka-docker/confluentinc-kafka-connect-jdbc-10.5.3/lib:/etc/kafka-connect/jars/
    environment:
      CONNECT_BOOTSTRAP_SERVERS: kafka-docker-kafka-1:9092
      CONNECT_REST_PORT: 8083
      CONNECT_GROUP_ID: "quickstart-avro"
      CONNECT_CONFIG_STORAGE_TOPIC: "quickstart-avro-config"
      CONNECT_OFFSET_STORAGE_TOPIC: "quickstart-avro-offsets"
      CONNECT_STATUS_STORAGE_TOPIC: "quickstart-avro-status"
      CONNECT_KEY_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_VALUE_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_INTERNAL_KEY_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_INTERNAL_VALUE_CONVERTER: "org.apache.kafka.connect.json.JsonConverter"
      CONNECT_REST_ADVERTISED_HOST_NAME: "localhost"
      # CONNECT_LOG4J_ROOT_LOGLEVEL: DEBUG
      CONNECT_PLUGIN_PATH: "/usr/share/java/kafka-connect-jdbc"
    networks:
      my-network:
        ipv4_address: 172.25.0.103

networks:
  my-network:
    name: artrend
```

- `docker-compose up -d`

5. 외부 서비스

1) 카카오

가. 애플리케이션 추가

기본 정보

수정

앱 아이콘	<div>APP</div>
앱 이름	아트렌드
사업자명	아트렌드

사용자가 카카오 로그인을 할 때 표시되는 정보입니다. 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

나. Redirect URI 설정

Redirect URI

삭제

수정

Redirect URI	<div>http://localhost:8080/auth-service/login/oauth2/code/kakao http://localhost:8080/auth-service/kakao/callback http://j7c104.p.ssafy.io:8080/auth-service/login/oauth2/code/kakao https://j7c104.p.ssafy.io:8080/auth-service/login/oauth2/code/kakao</div>
--------------	--

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

다. 동의 항목

개인정보			
항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	설정
프로필 사진	profile_image	● 사용 안함	설정
카카오계정(이메일)	account_email	● 사용 안함	설정

2) 구글

가. Redirect URI 설정

승인된 리디렉션 URI

웹 서버의 요청에 사용

URI 1 *

http://localhost:8080/auth-service/login/oauth2/code/google

URI 2 *

https://localhost:8080/auth-service/login/oauth2/code/google

URI 3 *








http://j7c104.p.ssafy.io:8080/auth-service/login/oauth2/code/google

URI 4 *

https://j7c104.p.ssafy.io:8080/auth-service/login/oauth2/code/google

[+ URI 추가](#)

나. 동의 항목

 필터 속성 이름 또는 값 입력 			
 API 	범위	사용자에게 표시되는 설명	
<input checked="" type="checkbox"/>	.../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인	
<input checked="" type="checkbox"/>	.../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기	
<input checked="" type="checkbox"/>	openid	Google에서 내 개인 정보를 나와 연결	
<input type="checkbox"/>	 BigQuery API	.../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account
<input type="checkbox"/>	 BigQuery API	.../auth/cloud-platform	Google Cloud 데이터 확인, 수정, 구성, 삭제 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/>	 BigQuery API	.../auth/bigquery .readonly	Google BigQuery에서 데이터를 봅니다.