exec

1

1. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정값, 버전(IDE버전 포함) 기재

- JVM : openjdk:11-jdk (back/demo/Dockerfile 참조)
- IntelliJ : 2022.2
- node.js : 16.16.0
- docker : 20.10.12
- webserver : nginx 1.23.1 (Docker)
  - docker를 통해 container로 띄움
  - docker run -itd --name webserver -v /home/ubuntu/docker-image/ssl:/ssl -p 443:443 -p 80:80 nginx

  - nginx container 내부 /etc/nginx/conf.d/deafault.conf 내부 변경

```
server {
    listen       443 ssl;
    server_name  i7d206.p.ssafy.io;

    ssl on;
    ssl_certificate /ssl/fullchain.pem;
    ssl_certificate_key /ssl/privkey.pem;
    ssl_prefer_server_ciphers on;

    #access_log  /var/log/nginx/host.access.log  main;
    #include /etc/nginx/conf.d/service-url.inc;

    root /usr/share/nginx/html/dist;

    location / {
        # root /usr/share/nginx/html/dist;
        index index.html index.htm;
        try_files $uri $uri/ @rewrite;
    }

    location @rewrite {
        rewrite ^(.+)$ /index.html last;
    }
```

- data base : mariadb 10.8.3 (Docker)
  - docker를 통해 container로 띄움
  - docker run -itd --name mariadb -e MYSQL_ROOT_PASSWORD=dearme -p 3306:3306 mariadb
  - mariadb container 내부 /etc/mysql/my.cnf 내용

```
[mariadbd]
skip-host-cache
skip-name-resolve

!includedir /etc/mysql/mariadb.conf.d/
!includedir /etc/mysql/conf.d/

lower_case_table_names = 1
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8
```

  - ‘dearme’ database 생성

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| dearme             |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

- create user ‘dearme’@’%’ identified by ‘dearme’;

```
MariaDB [(none)]> select host, user from mysql.user;
+-----------+-------------+
| Host      | User        |
+-----------+-------------+
| %         | dearme      |
| %         | root        |
| localhost | mariadb.sys |
| localhost | root        |
+-----------+-------------+
```

- grant all privileges on dearme.* to ‘dearme’@’%’ identified by ‘dearme’;

2. 빌드 시 사용되는 환경 변수 등의 주요 내용 상세 기재

- Dockerfile

```
FROM maven:3.8.6-openjdk-11 as 🔨 builder
COPY pom.xml .
COPY src ./src
RUN mvn -f pom.xml package

FROM openjdk:11-jdk
COPY --from=builder ./target/*.jar ./boot/app.jar
COPY --from=builder ./src ./src
RUN apt-get -y update
RUN apt-get -y upgrade
RUN apt-get install -y ffmpeg
EXPOSE 8080

ENTRYPOINT ["java","-jar", "-Dspring.profiles.active=deploy", "boot/app.jar" ]
```

- maven 3.8.6을 통해 빌드해준다.(mvn -f pox.xml package)

- 빌드한 파일을 openjdk:11-jdk 환경에서 java -jar -Dsrpong.profiles.active=deploy boot/app.jar를 통해 실행시켜준다.

- deploy.sh

```
docker build -t dearme_server .

DOCKER_APP_NAME=dearme_server

EXIST_BLUE=$(docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml ps | grep Up)

if [ -z "$EXIST_BLUE" ]; then
  echo "BLUE UP"
  docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml up -d
  sleep 20
  START_PORT=9090
  TERMINATE_PORT=9091
  docker exec webserver sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
  echo "nginx reload..."
  docker exec webserver service nginx reload
  sleep 5
  docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml down
else
  echo "GREEN UP"
  docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml up -d
  sleep 20
  START_PORT=9091
  TERMINATE_PORT=9090
  docker exec webserver sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
  echo "nginx reload..."
  docker exec webserver service nginx reload
  sleep 5
  docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml down
fi
```

  - Dockerfile을 통해 dearme_server라는 이름의 이미지로 빌드한다.
  - docker-compose를 이용해 dearme_server-blue라는 container가 돌아가는지 확인하고, 존재한다면 docker-compose를 이용해 dearme_server-green container를 올리고 nginx container 내부의 9090과 9091을 바꾼 후, nginx를 reload 해준다.
  - 만약 dearme_server-blue가 없다면 반대로 진행해준다.

3. 배포 시 특이사항 기재

- 배포시 deploy.sh를 실행시키면 Dockerfile과 docker-compose 파일을 사용하여 알아서 docker image를 만들고 docker-compose 파일을 이용하여 container를 띄운다.
  - 배포시 jenkins에서 deploy.sh를 실행시키기 때문에 jenkins는 사용자 권한이 있어야한다.

    - jenkins또한 docker로 띄웠기 때문에 jenkins container 내부에 docker, docker-compose 설치해주고 사용자 권한을 줘야한다.

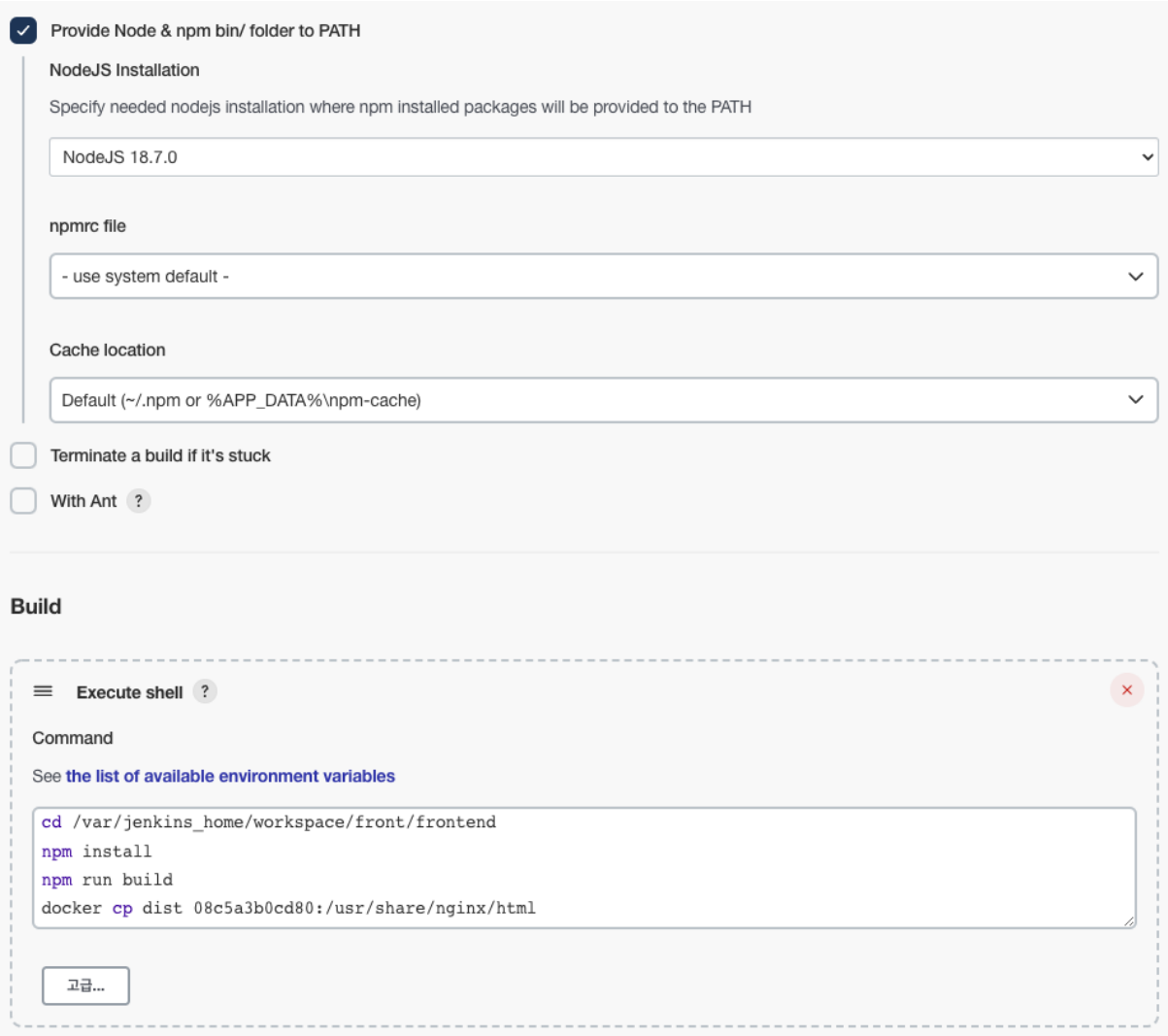
- 해당 container는 9090:8080 or 9091:8080으로 포트 설정이 되어있으며 nginx(Docker)를 reverse_proxy로 사용하여 https://i7d206.p.ssafy.io/api url로 들어온 모든 요청을 해당 container로 돌린다.
- 아무런 table이 없는 DB에 초기화 데이터를 사용하기위해선 application-deploy.properties 내부 설정을 변경해준다. (1회)

```
spring.application.active=deploy

server.ssl.enabled=true
server.ssl.key-store=classpath:key.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=dearme

spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.username=dearme
spring.datasource.password=dearme
spring.datasource.url=jdbc:mariadb://i7d206.p.ssafy.io:3306/dearme

spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
spring.jpa.show-sql=true

spring.jpa.defer-datasource-initialization=true
spring.sql.init.mode=always
spring.sql.init.data-locations=classpath:sql/data.sql

path.image=/image/
path.video=/video/

sentiment.id=w9jazjzk55
sentiment.key=fDUi38NcCgGHvIVgivrb7EbVuX7IxXMnYr9sxXjD

openvidu.openvidu_url=https://i7d206.p.ssafy.io:4443/
openvidu.password=dearme
```

```
message.access=TMT1IsuM3qkEm2yQn6XI
message.secret=x6qkoe05cswMJdyhFSv090qcNywkkG1qcTFYiE1r
message.url=/sms/v2/services/ncp:sms:kr:257491845770:deame/messages

server.servlet.context-path=/api
```

    - 1회만 해당 properties로 빌드 후 이후에는 원래대로 빌드해준다.

- 프론트 배포시 jenkins에서 제공해주는 nodejs 플러그인을 사용해서 발두 휴 nginx container 내부에 docker cp로 옮겨준다.



4. DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- back/demo/src/main/resources/application-deploy.properties