포팅 메뉴얼

버전

FrontEnd

- React : 18.2.0
- Node : 18.18.2(로컬, 테스트 환경) 18.18.1(배포 환경)
- TypeScript : 4.9.5
- Redux-toolkit : 1.9.7
- ESLint
- PortOne : 0.0.3

Backend

- Spring boot : 2.7.17
- java : 17
- IntelliJ : Ultimate 23.1
- Mysql : 8.0.34(로컬, 테스트 환경) 8.2.0(배포 환경)
- Redis : 3.0.504(로컬, 테스트 환경) 7.2.2(배포 환경)

AI

- Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-126-generic x86_64)
- Nvidia Geforce RTX 3080 *2
- RAM: 16GB * 8
- CPU: AMD Ryzen Threadripper 3970X 32-Core Processor

협업툴

- Gitlab
- Jira
- Notion

CI/CD

- Jenkins
- Docker
- Nginx
- AWS EC2 (Ubuntu 20.04.6 LTS)

수동 배포

Docker / Nginx / SSL

1. Nginx

```
# Nginx 설치
sudo apt install nginx

# Nginx 상태 확인
sudo systemctl status nginx

# Nginx 실행 시작/ 중지
```

```
sudo systemctl start nginx
sudo systemctl stop nginx

# Nginx 환경 설정
sudo vi /etc/nginx/sites-available/default.conf
```

2. Docker

```
# HTTPS를 통해 리포지토리를 사용할 수 있도록 패키지 인덱스를 업데이트 (apt)하고
# 패키지를 설치
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

# Docker의 공식 GPG 키를 추가
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# 리포지토리를 설정
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

-------------------------------------------------------------------------------
# 패키지 index를 업데이트
sudo apt-get update

# Docker Engine, containerd 및 Docker Compose를 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 이미지를 실행하여 Docker 엔진 설치가 성공했는지 확인
sudo docker run hello-world
docker -v
```

- Docker Network 설정
  - 같은 호스트내에서 실행 중인 컨테이너 간 연결을 할 수 있도록 돕는 논리적 개념

1. Docker network 생성 ⇒ `docker network create {네트워크 이름}`

3. SSL

```
# Let's Encrypt 설치
sudo apt-get install letsencrypt

# Certbot 설치
sudo apt-get install certbot python3-certbot-nginx

# Certbot 실행
sudo cerbot --nginx

# 필요한 사항 입력
# 약관동의, 이메일 입력
# 도메인 입력
```

Database

1. **Redis**

a. Docker 컨테이너를 위한 Redis 이미지 설치 ⇒ `docker pull redis`

b. Redis 환경 설정 파일 작성

```
# 외부 접속 허용
bind 0.0.0.0

# 포트 번호 변경
port {포트번호}
```

```
# 비밀번호 설정
requirepass {비밀번호}
```

- Redis Container 실행 ⇒ `docker run -d -p 6377:6377 --network {네트워크 이름} -v /redis/redis.conf:/etc/redis/redis.conf -e TZ=Asia/Seoul --name redis redis /etc/redis/redis.conf`
  - `docker run` → docker container 실행
  - `-d -p 6377:6377` → 포트번호 연결 ec2 내 포트번호 : 컨테이너 내 포트번호
  - `--network {네트워크 이름}` → docker network 연결
  - `-v /redis/redis.conf:/etc/redis/redis.conf` → 환경 설정 파일 마운트
  - `-e TZ=Asia/Seoul` → 시간 설정
  - `--name redis` → Container 이름 설정
  - `redis /etc/redis/redis.conf` → 띄울 image 설정, 실행 시킬 환경 설정 파일 위치
- 실행 확인 ⇒ `docker ps`

2. **Mysql**

a. Docker Container를 위한 Mysql 이미지 설치 ⇒ `docker pull mysql`

b. Mysql Container 실행 ⇒ `docker run -d -p 3306:3306 --network {네트워크 이름} -e MYSQL_ROOT_PASSWORD={ROOT PASSWORD} --name mysql mysql`

- `docker run` → docker container 실행
- `-d -p 6377:6377` → 포트번호 연결 ec2 내 포트번호 : 컨테이너 내 포트번호
- `--network {네트워크 이름}` → docker network 연결
- `-e MYSQL_ROOT_PASSWORD={ROOT PASSWORD}` → ROOT_PASSSWORD 설정
- `--name mysql mysql` → container 이름 설정, 띄울 image 설정\

c. 데이터베이스 생성 및 권한 설정

- `docker exec -it {container 이름} mysql -root -p{ROOT_PASSSWORD}` → container 접속
- `create database 'database_name';` → Database 생성
- `create user 'user_name'@'IP주소' identified by 'user_password';` → user 생성
- `grant all privileges on 'database_name'.* to 'user_name'@'IP주소';` → 권한 부여
- `flush privileges;` → 변경 사항 적용

Backend & FrontEnd

1. Git clone

```
git clone
```

2. FrontEnd 빌드 & 배포

```
# 폴더 이동
cd frontend

# 패키지 설치
npm install

# 프로젝트 빌드
npm run build

# docker image 생성
docker build -t frontend .

# docker container 실행
docker run
-d -p 3000:3000
-e TZ=Asia/Seoul
```

```
--network {네트워크 이름}
--name frontend frontend
```

3. Backend 빌드 & 배포

```
# 폴더 이동
cd /backend/Ddobak

# 권한 생성
chmod +x gradlew

# 프로젝트 빌드
./gradlew clean bootjar

# dokcer image 생성
docker build -t backend .

# docker container 실행
docker run
-d -p 7077:7077
--network {네트워크 이름}
-e EMAIL_SERVICE_ID={구글 이메일 서비스 주소}
-e EMAIL_SERVICE_PASSWORD={구글 이메일 서비스 비밀 키}
-e JWT_SECRET_KEY={JWT 생성 비밀 키}
-e MYSQL_PASSWORD={로컬 데이터베이스 비밀번호}
-e MYSQL_USER={로컬 데이터베이스 계정}
-e REDIS_PASSWORD={서버 데이터베이스 비밀번호}
-e S3_ACCESS_KEY={AWS S3 ACCESS KEY}
-e S3_BUCKET_NAME={AWS bucket name}
-e S3_REGION={AWS region}
-e S3_SECRET_KEY={AWS secret key}
-e SERVER_MYSQL_USERNAME={서버 데이터베이스 계정}
-e SERVER_MYSQL_PASSWORD={서버 데이터베이스 비밀번호}
-e TZ=Asia/Seoul
--name backend backend
```

Nginx

1. nginx 설정 포트 포워딩

- ec2 접속
- `cd /etc/nginx/sites-available` → 설정 파일 위치로 이동
- `sudo vi default` → 설정 파일 편집, 아래의 설정으로 편집

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;


        root /var/www/html;

        # Add index.php to the list if you are using PHP

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }


}
server {

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
        server_name ddobak.com; # managed by Certbot


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
```

```
                # try_files $uri $uri/ =404;
                proxy_pass http://localhost:3000;
        }
        location /swagger-ui {
                proxy_pass http://localhost:7077;
        }

        location /api {
                proxy_pass http://localhost:7077;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/ddobak.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/ddobak.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot



}
server {
    if ($host = ddobak.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name ddobak.com;
    return 404; # managed by Certbot


}
server {

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name ddobak.com; # managed by Certbot


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/ddobak.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/ddobak.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    listen 80;
    listen [::]:80;
    server_name k9c208.p.ssafy.io;

    location / {
        return 301 https://ddobak.com$request_uri;
    }
}
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name k9c208.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k9c208.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k9c208.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        return 301 https://ddobak.com$request_uri;
    }
}

server {
        listen 8090 ssl;
        server_name ddobak.com;
```

```
        ssl_certificate /etc/letsencrypt/live/ddobak.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/ddobak.com/privkey.pem;

        location / {
                proxy_pass http://localhost:8080;
        }
}
server {
    listen 8090 ssl;
    server_name k9c208.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k9c208.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k9c208.p.ssafy.io/privkey.pem;

    location / {
        proxy_pass http://localhost:8080;
    }
}
```

AI 서버

1. Nvidia-driver 설치

```
# 지원되는 그래픽 드라이버 확인 -> recommended 설치 권장
ubuntu-drivers devices

# 설치 가능한 nvidia driver 디바이스 목록이 출력.

# 아래 링크에서 설치하고자 하는 CUDA 버전과 그에 따른 nvidia driver 리스트를 확인

# https://docs.nvidia.com/deploy/cuda-compatibility/index.html#binary-compatibility__table-toolkit-driver
# → CUDA 11.8을 설치할 거기 때문에 470버전으로 설치 해준다.
```

```
# 권장 드라이버 버전 설치
sudo apt install nvidia-driver-470

# 재시작
reboot

# 설치 확인
nvidia-smi
```

2. CUDA 설치

- CUDA Toolkit Archive에서 `nvidia-smi` 명령어를 통해 확인한 CUDA Version 설치
  - https://developer.nvidia.com/cuda-toolkit-archive

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1
sudo dpkg -i cuda-repo-ubuntu1804-11-8-local_11.8.0-520.61.05-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu1804-11-8-local/cuda-*-keyring.gpg /usr/share/keyrings/
sudo apt-get update
sudo apt-get -y install cuda
```

- 환경 변수 추가

```
export PATH=$PATH:/usr/local/cuda-11.8/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.8/lib64
export CUDADIR=/usr/local/cuda-11.8
```

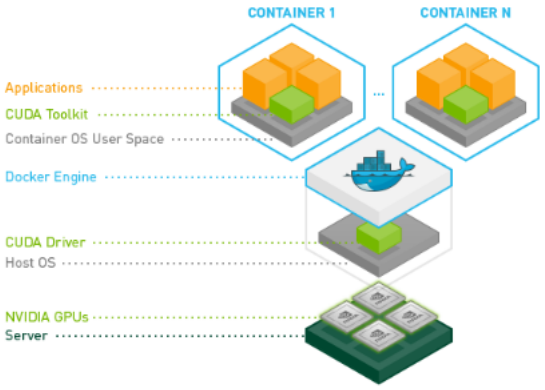
```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Nov__3_21:07:56_CDT_2017
Cuda compilation tools, release 9.1, V9.1.85
```

```
sudo apt install nvidia-cuda-toolkit
```

```
nvcc -V
```

3. NVIDIA Container ToolKit 설치
   - 도커 환경에서 GPU를 사용하기 위해 NVIDIA Container Toolkit 설치를 해준다.

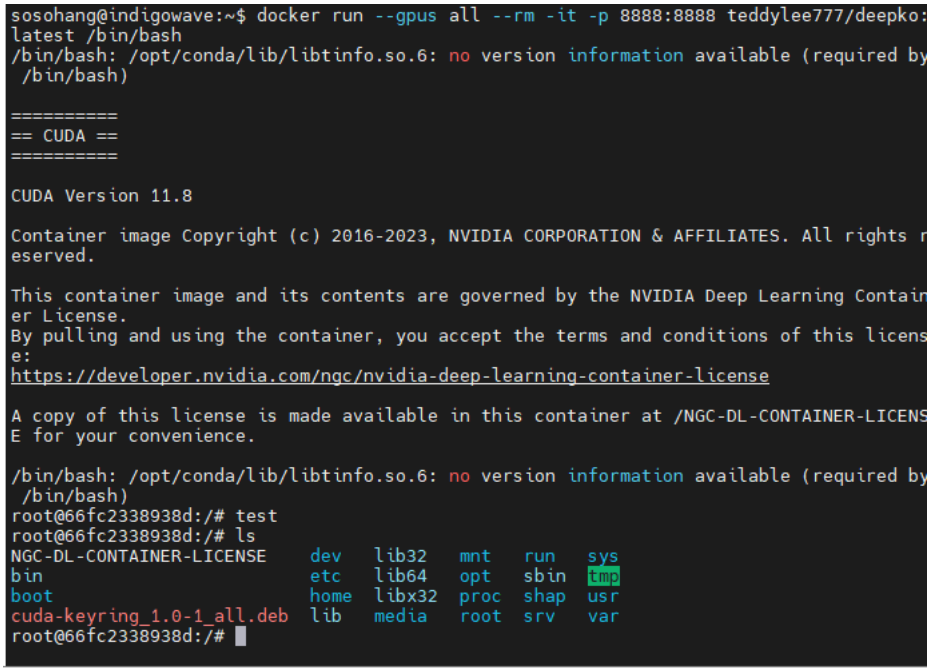
     https://github.com/NVIDIA/nvidia-docker#nvidia-container-toolkit




```
# 레포지토리 등록
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolki
  && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list \
  && \
    sudo apt-get update

# 패키지 설치
sudo apt-get install -y nvidia-container-toolkit
```

4. NVIDIA Docker Repository 설정

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docke

# 패키지 업데이트 및 nvidia-docker2 설치:
sudo apt-get update
sudo apt-get install -y nvidia-docker2

sudo systemctl restart docker
docker --version
nvidia-docker --version
```


sosohang@indigowave:~$ docker run --gpus all --rm -it -p 8888:8888 teddylee777/deepko:
latest /bin/bash
/bin/bash: /opt/conda/lib/libtinfo.so.6: no version information available (required by
/bin/bash)
==========
== CUDA ==
==========
CUDA Version 11.8
Container image Copyright (c) 2016-2023, NVIDIA CORPORATION & AFFILIATES. All rights r
eserved.
This container image and its contents are governed by the NVIDIA Deep Learning Contain
er License.
By pulling and using the container, you accept the terms and conditions of this licens
e:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license
A copy of this license is made available in this container at /NGC-DL-CONTAINER-LICENS
E for your convenience.
/bin/bash: /opt/conda/lib/libtinfo.so.6: no version information available (required by
/bin/bash)
root@66fc2338938d:/# test
root@66fc2338938d:/# ls
NGC-DL-CONTAINER-LICENSE
bin
boot
cuda-keyring_1.0-1_all.deb
dev
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
shap
srv
sys
tmp
usr
var
root@66fc2338938d:/#