포팅 매뉴얼

버전

BACKEND

Java 17

SpringBoot 3.1.1

querydsl-jpa 5.0.0

Gradle

FRONTEND

react 18.2.0

react-redux 8.1.1

react-router-dom 6.14.1

axios 1.4.0

bootstrap 5.3.0

env-cmd 10.1.0

ETC

nginx 1.25.1

redis:latest

docker

빌드

1. docker 설치

```
sudo apt-get update && upgrade
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/li
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker $USER
```

2. nginx, certbot, let’s encrypt 인증서 발급

docker-compose.yml

```
version: '3'
services:
  nginx:
    image: nginx:latest
    restart: unless-stopped
    volumes:
      - ./conf/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
```

```
    ports:
      - 80:80
      - 443:443
  certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
```

conf/nginx.conf 생성

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

     server_name unofficial.kr;

     location /.well-known/acme-challenge/ {
             allow all;
             root /var/www/certbot;
     }
}
```

docker-compose 실행

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

인증서 발급받는 스크립트를 다운로드하고 도메인, 이메일 주소, 디렉터리를 변경합니다.

```
curl -L <https://raw.githubusercontent.com/wmnnd/nginx-certbot/master/init-letsencrypt.sh> > init-letsencrypt.sh
chmod +x init-letsencrypt.sh
vi init-letsencrypt.sh // 도메인, 이메일, 디렉토리 수정
sudo ./init-letsencrypt.sh // 인증서 발급
```

nginx.conf 수정

```
events {
     worker_connections 2048;
}

http {
     include /etc/nginx/mime.types;
     limit_req_zone $binary_remote_addr zone=ddos_req:10m rate=20r/s;

     upstream spring {
          server 172.18.0.50:8080;
          server 172.18.0.51:8080;
     }

     client_max_body_size 20M;

     server {
          listen 80;
          server_name vidu.unofficial.kr;
          server_tokens off;

          location /.well-known/acme-challenge/ {
                  root /var/www/certbot;
          }
          location / {
                  return 301 https://vidu.unofficial.kr$request_uri;
          }
     }

     server {
          listen 80;
          server_name dev.unofficial.kr;
          server_tokens off;

          location /.well-known/acme-challenge/ {
                  root /var/www/certbot;
```

```
    }
    location / {
            return 301 https://dev.unofficial.kr$request_uri;
    }
}

server {
    listen 80;
    server_name container.unofficial.kr;
    server_tokens off;

    location /.well-known/acme-challenge/ {
            root /var/www/certbot;
    }
    location / {
            return 301 https://container.unofficial.kr$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name container.unofficial.kr;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/unofficial.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/unofficial.kr/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
             proxy_pass https://172.18.0.5:9443;
             proxy_set_header     Host            $http_host;
             proxy_set_header     X-Real-IP       $remote_addr;
             proxy_set_header     X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
        listen 443 ssl;
        server_name vidu.unofficial.kr;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/unofficial.kr/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/unofficial.kr/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {
                proxy_pass http://172.18.0.4:4443;
                proxy_set_header   Host    $http_host;
                proxy_set_header   X-Real-IP       $remote_addr;
                proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        }
}

server {
    listen 80;
    server_name www.unofficial.kr unofficial.kr;
    server_tokens off;

    location /.well-known/acme-challenge/ {
            root /var/www/certbot;
    }
    location / {
            return 301 https://unofficial.kr$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name www.unofficial.kr;
    server_tokens off;

    location / {
             return 301 https://unofficial.kr$request_uri;
    }
}

server {
        listen 443 ssl;
        server_name dev.unofficial.kr;
        server_tokens off;

        ssl_certificate /home/letsencrypt/live/unofficial.kr/fullchain.pem;
        ssl_certificate_key /home/letsencrypt/live/unofficial.kr/privkey.pem;

        location /api {
```

```
                    proxy_pass http://172.18.0.100:8080;
                    proxy_set_header   Host    $http_host;
                    proxy_set_header   X-Real-IP       $remote_addr;
                    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            }
            location /v3 {
                    proxy_pass http://172.18.0.100:8080;
                    proxy_set_header   Host    $http_host;
                    proxy_set_header   X-Real-IP       $remote_addr;
                    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            }
            location / {
                    proxy_pass http://172.18.0.101;
                    proxy_set_header   Host    $http_host;
                    proxy_set_header   X-Real-IP       $remote_addr;
                    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            }
    }


    server {
         listen 443 ssl;
         server_name unofficial.kr;
         server_tokens off;

         ssl_certificate /etc/letsencrypt/live/unofficial.kr/fullchain.pem;
         ssl_certificate_key /etc/letsencrypt/live/unofficial.kr/privkey.pem;
         include /etc/letsencrypt/options-ssl-nginx.conf;
         ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

         location /api {
                 limit_req zone=ddos_req burst=5;
                 proxy_pass http://spring;
                 proxy_set_header     Host            $http_host;
                 proxy_set_header     X-Real-IP       $remote_addr;
                 proxy_set_header     X-Forwarded-For $proxy_add_x_forwarded_for;
         }

         location / {
                 proxy_pass http://172.18.0.52;
                 proxy_set_header   Host    $http_host;
                 proxy_set_header   X-Real-IP       $remote_addr;
                 proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
         }
    }
}
```

docker-compose.yml 수정

```
version: '3'

services:
  nginx:
    image: nginx:1.15-alpine
    restart: unless-stopped
    volumes:
      - ./data/nginx:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
    command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx -g \\"daemon off;\\"'"
  certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"
```

3. MySql 설치 및 세팅

QA

```
docker run --name QA-mysql-container \
      -e MYSQL_ROOT_PASSWORD=패스워드\
```

```
      -v /var/lib/docker/volumes/QAmysql/_data:/var/lib/mysql \
      --network ubuntu_default --ip 172.18.0.6 \
      -d mysql:8.0.34
```

ssafy_qa_db 스키마 생성

Product

```
docker run --name Prod-mysql-container \
      -e MYSQL_ROOT_PASSWORD=패스워드\
      -v /var/lib/docker/volumes/ProdMysql/_data:/var/lib/mysql \
      --network ubuntu_default --ip 172.18.0.60 \
      -d mysql:8.0.34
```

ssafy_web_db 스키마 생성

4. Redis 세팅

```
docker run --name redis --network ubuntu_default --ip 172.18.0.70 -d redis redis-server --save 60 1 --loglevel warning
```

```
docker run --name QA-redis --network ubuntu_default --ip 172.18.0.8 -d redis redis-server --save 60 1 --loglevel warning
```

5. gitlab-runner 설치 및 세팅

```
sudo curl -L --output /usr/local/bin/gitlab-runner "https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner
sudo chmod +x /usr/local/bin/gitlab-runner
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash
# Optional sudo rm /etc/systemd/system/gitlab-runner.service
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
```

config.toml 설정

```
concurrent = 1
check_interval = 0

[[runners]]
  name = "ip-172-26-14-68"
  url = "https://lab.ssafy.com"
  token = "glrt-fQ-Vx2wzVk6QRgNNtoy6"
  executor = "shell"
  [runners.cache]
```

gitlab-ci.yml

```
stages: # List of stages for jobs, and their order of execution
  - test
  - build
  - libs
  - deploy
  - finish

test: # This job runs in the build stage, which runs first.
  image: openjdk:17.0.1-oraclelinux8
  stage: test
  script:
    - chmod -R 775 server/
    - cd ./server/Strange505
    - ./gradlew test
```

```
build: # This job runs in the test stage.
  image: openjdk:17.0.1-oraclelinux8
  stage: build # It only starts when the job in the build stage completes successfully.
  variables:
    IMAGE_NAME: strangedev505/springboot-app
  script:
    - chmod -R 775 server/
    - cd ./server/Strange505
    - ./gradlew clean
    - ./gradlew build
    - docker build -t $IMAGE_NAME --build-arg env=serve .
  only:
    - master

npm:
  stage: libs
  cache:
    paths:
      - frontend/node_modules
  script:
    - chmod -R 775 frontend/
    - cd frontend
    - npm install

deploy-was1: # 스프링 백엔드 1.
  stage: deploy
  tags:
    - deployer
  script:
    - docker stop springboot-app || true
    - docker rm springboot-app || true
    - docker run -d --restart always --network ubuntu_default --ip 172.18.0.50 -e TZ=Asia/Seoul --name springboot-app strangedev505
  when: on_success
  only:
    - master

deploy-was2: # 스프링 백엔드 2
  stage: deploy
  tags:
    - deployer
  script:
    - docker stop springboot-app2 || true
    - docker rm springboot-app2 || true
    - docker run -d --restart always --network ubuntu_default --ip 172.18.0.51 -e TZ=Asia/Seoul --name springboot-app2 strangedev50
  when: on_success
  only:
    - master

deploy-react: # 리액트 컨테이너
  stage: deploy
  variables:
    IMAGE_NAME: strangedev505/react
    NAME: react
  tags:
    - deployer
  cache:
    paths:
      - dist
  script:
    - chmod -R 775 frontend/
    - cd frontend
    - npm install
    - npm install --save-dev @babel/plugin-proposal-private-property-in-object
    - npm run build
    - docker build -t $IMAGE_NAME --build-arg env=prod .
    - docker stop $NAME || true
    - docker rm $NAME || true
    - docker run -d --restart always --network ubuntu_default --ip 172.18.0.52 -e TZ=Asia/Seoul --name $NAME $IMAGE_NAME
  only:
    - master

finish:
  stage: finish
  script:
    - docker rmi $(docker images -f "dangling=true" -q)
    - docker restart ubuntu_nginx_1
  only:
    - master

build-dev:
  image: openjdk:17.0.1-oraclelinux8
  stage: build # It only starts when the job in the build stage completes successfully.
  variables:
    IMAGE_NAME: strangedev505/springboot-dev
  script:
    - chmod -R 775 server/
```

```
    - cd ./server/Strange505
    - ./gradlew clean
    - ./gradlew build
    - docker build -t $IMAGE_NAME --build-arg env=qa .
  only:
    - develop

deploy-was1-dev: # 스프링 qa 백엔드
  stage: deploy
  tags:
    - deployer
  script:
    - docker stop springboot-dev || true
    - docker rm springboot-dev || true
    - docker run -d --restart always --network ubuntu_default --ip 172.18.0.100 -e TZ=Asia/Seoul --name springboot-dev strangedev50
  when: on_success
  only:
    - develop

deploy-react-dev:  # 리액트 qa 컨테이너
  stage: deploy
  variables:
    IMAGE_NAME: strangedev505/react-dev
    NAME: react-dev
  tags:
    - deployer
  cache:
    paths:
      - dist
  script:
    - chmod -R 775 frontend/
    - cd frontend
    - npm install
    - npm install --save-dev @babel/plugin-proposal-private-property-in-object
    - npm run build:dev
    - docker build -t $IMAGE_NAME --build-arg env=qa .
    - docker stop $NAME || true
    - docker rm $NAME || true
    - docker run -d --restart always --network ubuntu_default --ip 172.18.0.101 -e TZ=Asia/Seoul --name $NAME $IMAGE_NAME
  only:
    - develop

finish-dev:
  stage: finish
  script:
    - docker restart ubuntu_nginx_1
  only:
    - develop
```

5. develop이나 master 변경되면 자동빌드

외부 API

welstory API 웰스토리 계정 필요

SMTP 메일 전송을 위한 구글 계정 필요