포팅 매뉴얼

1. 개발 환경

1.1 개발 환경

- JAVA 11
- SpringBoot : 2.7.15
- Gradle : 8.2.1
- Node.js : 18.16.1
- mysql : 8.0.32

1.2. gitignore 환경 변수

- `Spring` - `/BE/hpdp` - `application-prod.yml`

```
# server port
server:
  address: localhost

spring:
  config:
    activate:
      on-profile: prod

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://{DB url}/{DB 이름}?useSSL=false&characterEncoding=UTF-8&serverTimezone=UTC&allowPublicKeyRetrieval=true
    username: {db ID}
    password: {db 비밀번호)

  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        show_sql: true
        default_batch_fetch_size: 100
    open-in-view: false

  # Redis
  redis:
    port: {port 번호}
    host: {DB url}
    password: {DB password}

  thymeleaf:
    check-template-location: true
    suffix: .html
    mode: HTML
    cache: false

jwt:
  secret: {jwt 비밀키}

cloud:
  aws:
    credentials:
      access-key: {aws accessKey}
      secret-key: {aws secret}
    s3:
      bucket: {S3 이름}
    region:
      static: {S3 지역}
    stack:
      auto: false
```

```
ethereum:
  rpc-url: https://j9c110.p.ssafy.io/bc
  password: 1q2w3e4r!

privatekey:
  admin : bc3724072b1bbbaae643d07e2c39f1ab1cf9c003befc91ff3e87b6c62908e8db
  bepo : 912b96488ad18f82928d5cc9c3d6305db09c4e6b5f79a5518a241204926fa973
address:
  admin: "0x9A6D9dB08f536fd90DD4f77Ac79e17FA6B9c1E6a"
  token: "0x54B2008EC541Bd54e92f0901F7D2162e6D688FA0"
  funding: "0xC68E1341cd4334dd7be7831350C440E7548b2fA2"
```

- `FastAPI` - `/BE/hpdp_AI` - `/app/.env`

```
OPENAI_API_KEY={openai key}
```

1.3 외부 서비스

- Kakaopay API

2. 배포

2.0 Docker 설치

2.1 geth 배포

- Go 설치하기

```
sudo apt update
sudo apt install golang

sudo apt install -y libgmp3-dev tree make build-essential
```

- Go 버전 업그레이드

```
git clone https://github.com/udhos/update-golang
cd update-golang
sudo ./update-golang.s
```

- geth 다운로드

```
cd ~
mkdir Ethereum
cd Ethereum

git clone https://github.com/ethereum/go-ethereum
sudo apt-get update
sudo apt-get -y upgrade


cd go-ethereum
make geth
```

- geth 실행

```
cd go-ethereum

cd ./build/bin

./geth version

./geth
```

- geth 환경 변수 설정

```
pwd
결과로 나온 값 복사

vi ~/.bash_profile

export PATH=$PATH:(아까 나온값)

source ~/.bash_profile
```

- 방화벽 열기

```
sudo ufw allow ssh
sudo ufw allow 30305/tcp
sudo ufw allow 30305/udp
sudo ufw allow 30306/tcp
sudo ufw allow 30306/udp
sudo ufw allow 30307/tcp
sudo ufw allow 30307/udp
sudo ufw allow 8551/tcp
sudo ufw allow 8551/udp
sudo ufw allow 8552/tcp
sudo ufw allow 8552/udp
sudo ufw allow 3334/tcp
sudo ufw allow 3334/udp
sudo ufw enable
```

- 노드 실행
  - node1.sh

```
geth \
--datadir node1 \
--port 30306 \
--bootnodes "enode://db07c5c3afdfcf6837ae1110d65ecb4dc266603e7d98d5bb9310aa235fff6c0e11e7166de6df4cbbe414aacecf8688bcef4691c92f179
--networkid 12345 \
--unlock 0x9a6d9db08f536fd90dd4f77ac79e17fa6b9c1e6a,0x645882bfad675cb914d38f2d77461ca1cebbc58a,0x879788d3c7bf2161e0696146cfdcfc77c
--password node1/password.txt \
--authrpc.port 8551 \
--ipcpath node1.ipc \
--allow-insecure-unlock \
--http \
--http.port 3334 \
--http.addr 127.0.0.1 \
--http.api "miner,admin,personal,eth,net,web3" \
--http.corsdomain "*" \
--http.vhosts "*" \
--mine \
--miner.etherbase=0x9a6d9db08f536fd90dd4f77ac79e17fa6b9c1e6a \
console
```

  - node2.sh

```
geth \
--datadir node2 \
--port 30307 \
--bootnodes "enode://db07c5c3afdfcf6837ae1110d65ecb4dc266603e7d98d5bb9310aa235fff6c0e11e7166de6df4cbbe414aacecf8688bcef4691c92f179
--networkid 12345 \
--unlock 0xdc7b15b82d27ee3a6df4d7e043f4b463bc0a7a33 \
--password node2/password.txt \
--authrpc.port 8552 \
--ipcpath node2.ipc \
```

```
bootnode -nodekey boot.key -addr :30305

./node1.sh

./node2.sh
```

2.2 DB 배포

- **git clone**

```
git clone https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22C110.git
```

- **redis 배포**
  - 백엔드 배포 전에 배포 필수

```
sudo docker run --rm -d -p 6379:6379 --name redis_db {IMAGE ID}
```

- **mysql 배포**
  - 백엔드 배포 전에 배포 필수(db 설정 추가 필요)

```
docker run --name mysql_db -e MYSQL_ROOT_PASSWORD={mysql pw} -d -p 3305:3306 mysql:latest
```

2.3 백엔드 빌드 및 배포

- `/BE/hpdp` 폴더에 `Dockerfile` 위치

```
# Dockerfile

FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} springboot.jar
ENV USE_PROFILE prod
ENTRYPOINT ["java", "-Dspring.profiles.active=${USE_PROFILE}", "-jar", "springboot.jar"]
```

- 프로젝트 내 `/BE/hpdp` 폴더에서 다음 명령어 실행

```
# gradle build
gradle clean bootjar

# docker image build
sudo docker build -t {docker repository}:backend-0.1 .

# docker image push
docker push {docker repository}:backend-0.1
```

- `/BE/hpdp_AI` 폴더에 `Dockerfile` 위치

```
FROM python:3.10

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

RUN pip install BeautifulSoup4 openai load_dotenv uvicorn fastapi requests

WORKDIR /code

COPY ./app /code/app

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

- `/BE/hpdp_AI` 폴더에 `requirements.txt` 위치

```
fastapi>=0.68.0,<0.69.0
pydantic>=1.8.0,<2.0.0
uvicorn>=0.15.0,<0.16.0
```

- 프로젝트 내 `/BE/hpdp_AI` 폴더에서 다음 명령어 실행

```
# docker image build
sudo docker build -t {docker repository}:backendai-0.1 .
```

```
# docker image push
docker push {docker repository}:backendai-0.1
```

2.4 프론트엔드 빌드 및 배포

- `/FE` 폴더에 `Dockerfile`, `nginx.conf` 위치

```
FROM nginx
WORKDIR /app
RUN mkdir ./build
ADD ./build ./build
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

```
server {
    listen 3000;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

- 프로젝트 내 `/FE` 폴더에서 다음 명령어 실행

```
# module 설치
npm install
npm run sass

# npm build
npm run build

# docker image build
sudo docker build -t {docker repository}:frontend-0.1 .

# docker image push
docker push {docker repository}:frontend-0.1
```

2.5 Nginx 설정

- Nginx 설치

```
sudo apt-get install nginx
```

- SSL 인증서 발급

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

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

# Certbot 동작
sudo certbot --nginx
```

- `/etc/nginx/sites-available` 에 `hpdp.conf` 작성

```
server {
        location / {
                proxy_pass http://127.0.0.1:3000;
                proxy_buffer_size          128k;
                proxy_buffers              4 256k;
                proxy_busy_buffers_size    256k;

                proxy_set_header X-Real-IP $remote_addr;
```

```
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location /api {
                proxy_pass http://127.0.0.1:8080;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header Connection '';
                proxy_http_version 1.1;
                proxy_buffering off;
                chunked_transfer_encoding off;
        }
        location /articles {
                proxy_pass http://127.0.0.1:8000;

                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }
        location /bc {
                proxy_pass http://127.0.0.1:3334/;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header X-NginX-Proxy true;
        }

#       listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        server_name j9c110.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j9c110.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j9c110.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
        if ($host = j9c110.p.ssafy.io) {
                return 301 https://$host$request_uri;
        }

        listen 80;
#       listen [::]:80;
        server_name j9c110.p.ssafy.io;
        return 404;
```

- 설정 적용 및 Nginx 실행

```
# 심볼릭 링크 연결
sudo ln -s /etc/nginx/sites-available/hpdp.conf /etc/nginx/sites-enabled/hpdp.conf
# 재실행 -> hpdp.conf 파일 변경 시 재실행 해야함!!
sudo systemctl restart nginx

# 심볼릭 링크 제거(필요할 때 사용)
sudo rm /etc/nginx/sites-enabled/hpdp.conf
```

2.6 docker compose 설정

- `ubuntu` 내에 `docker-compose` 밑에 `docker-compose.yml` 위치

```
version: '3'

services:
  backend:
    container_name: backend
    restart: on-failure
    image: {backend image 위치}
    expose:
      - 8080
    ports:
      - "8080:8080"

  frontend:
```

```
    container_name: frontend
    image: {frontend image 위치}
    expose:
      - 3000
    ports:
      - "3000:3000"

  backendai:
    container_name: backendai
    image: {backendai image 위치}
    command: uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
    ports:
      - "8000:8000"
```

- `ubuntu` 내에 `/docker-compose` 폴더에서 다음 명령 실행

```
docker-compose up -d
```