빌드 및 배포 방법

버전

BACKEND(동화)

- Gradle : 7.6.1
- IntelliJ : 2022.3.1
- Spring
  - Spring Boot : 2.7.9
  - Spring Security : 2.7.9
  - Spring Data JPA : 2.7.8
  - Spring Cloud Starter AWS : 2.2.1
- Swagger3 : 3.0.0
- QueryDSL 5.0
- Json Web Token : 0.9.1
- jdk : openjdk 1.8.0.332
- MariaDB : 10.6.5
- lombok 1.18.26

FRONTEND(동화)

- npm : v9.2.0
- yarn : v.1.22.19
- React.js : v18.2.0
- zustand : v.4.3.6
- react-query : v4.26.1
- prettier : v.2.8.4
- web3 : v.1.9.0
- axios : v1.3.4
- react-cookie : v4.1.1
- react-pageflip : v2.0.3
- react-slick : v0.29.0
- slick-carousel : v1.8.1
- swiper : v9.2.0
- react-flip-clock-countdown : v.1.4.0
- emotion : v11.10.6

- styled-components : v.5.3.9
- twin.macro : v.3.1.0
- tailwindcss : v.3.2.7
- material-ui : v.5.11.14

BACKEND(지갑)

- Node.js : v18.14.2
- npm : v 9.5.0
- Express.js : v 4.18.2
- CORS : v 2.8.5
- Crypto-js : v 4.1.1
- Web3.js : v 1.9.0
- Ipfs-Http-Client : v 60.0.0
- Axios : v 1.3.4
- Multer : 1.4.5-lts.1

FRONTEND(지갑)

- Vue.js : v 3.2.47
- Vuex : v 4.0.0
- npm : v 9.5.0
- Axios : v 1.3.4
- Bootstrap : v 5.2.3

BLOCKCHAIN

- Solidity
- Remix.ide

SERVER

- AWS
  - S3
  - CloudFront
  - EC2
    - 플랫폼: Ubuntu 20.04.5 LTS
- Docker : 23.0.1

- Nginx : 1.18.0(Ubuntu)
- Jenkins : 2.375.3
- Jupyter Notebook : 6.5.2
- IPFS : 0.19.0

빌드&배포

본 프로젝트의 빌드와 배포는 Jenkins(Docker Container)와 Docker를 통해 이루어져 있으며, Git Lab과 Jenkins가 연동되어있음을 전제로 한다.

또한, Docker 명령어 수행을 위해 Jenkins Container 실행시

-v /home/jenkins:/var/jenkins_home 옵션과

-v /var/run/docker.sock:/var/run/docker.sock 옵션으로

호스트와 컨테이너를 볼륨 매핑 해야한다.

BACKEND(동화)

주의사항

gitlab에는 application-dev.yml파일이 포함되어있지 않으므로 다음의 파일을 세팅해야 빌드가 가능하다.

```
spring:
  datasource:
    url: jdbc:mariadb:사용할 db주소?characterEncoding=UTF-8
    driver-class-name: org.mariadb.jdbc.Driver
    username: 유저명
    password: 비밀키
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true

  servlet:
    multipart:
      max-file-size: 20MB
      max-request-size: 25MB

  redis:
    host: 호스트 url
    port: 사용 포트
    password: 비밀키

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 카카오 id
            client-secret: 카카오 비밀키
            redirect-uri: [로그인 리다이렉트 url 주소](로그인 리다이렉트 url 주소)
            authorization-grant-type: authorization_code
            client-authentication-method: POST
            client-name: Kakao
            scope:
              - profile_nickname
```

```
          google:
            client_id: 구글 id
            client_secret: 구글 비밀키
            redirect-uri: <로그인 리다이렉트 url 주소>
            scope: profile

        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

          google:
            authorization-uri: https://accounts.google.com/o/oauth2/v2/auth
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://www.googleapis.com/oauth2/v3/userinfo

app:
  auth:
    tokenSecret: 토큰 비밀키
    tokenExpirationMsec: 6000000
    refreshTokenExpiry: 1209600000
  key:
    aesSecret: aes 비밀키
    iv : aes에 사용할 is키
  oauth2:
    # After successfully authenticating with the OAuth2 Provider,
    # we'll be generating an auth token for the user and sending the token to the
    # redirectUri mentioned by the client in the /oauth2/authorize request.
    # We're not using cookies because they won't work well in mobile clients.
    authorizedRedirectUris:
      - [로그인 리다이렉트 url 주소](로그인 리다이렉트 url 주소)

cloud:
  aws:
    credentials:
      access-key: aws 엑세스 키
      secret-key: aws 시크릿 키
    s3:
      bucket: bucket 명
    region:
      static: 지역 명
    stack:
      auto: false
```

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/backend/

혹은 clone 받은 S08P22A308/backend/에

Dockerfile을 생성한다.

```
FROM openjdk:8-jdk

#JAR_FILE 변수 정의 -> 기본적으로 jar file이 2개이기 때문에 이름을 특정해야함
ARG JAR_FILE=./build/libs/mongttang-0.0.1-SNAPSHOT.jar

#JAR 파일 메인 디렉토리에 복사
COPY ${JAR_FILE} app.jar

#시스템 진입점 정의.
ENTRYPOINT ["java","-jar","/app.jar"]
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/mongttang_spring/backend
chmod +x gradlew
```

```
./gradlew build
docker build -t mongttang-spring .
if [ $( docker ps -a | grep mongttang-spring | wc -l ) -gt 0 ]; then
echo "mongttang-spring container exists"
echo "delete container"
docker stop mongttang-spring
docker rm mongttang-spring
else
echo "mongttang-spring container does not exist"
fi
docker run -p 8308:8308 --name mongttang-spring -d mongttang-spring
```

FRONTEND(동화)

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/frontend

혹은 clone 받은 S08P22A308/frontend에

Dockerfile과 default.conf를 생성한다.

Dockerfile

```
FROM node as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build
FROM nginx:1.13.9-alpine

# nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm -rf /etc/nginx/conf.d
COPY default.conf /etc/nginx/conf.d/default.conf

# 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/build /usr/share/nginx/html

# 80포트 오픈하고 nginx 실행
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

default.conf

```
server {
  listen 80;
  location / {
    root   /usr/share/nginx/html;
    index  index.html;
    try_files $uri $uri/ /index.html;
  }
}
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/mongttang_react/frontend
docker build -t mongttang-react .
if [ $( docker ps -a | grep mongttang-react | wc -l ) -gt 0 ]; then
echo "mongttang-react container exists"
echo "delete container"
docker stop mongttang-react
docker rm mongttang-react
else
echo "mongttang-react container does not exist"
fi
docker run -p 3000:80 --name mongttang-react -d mongttang-react
```

BACKEND(지갑)

주의사항

gitlab에는 index.js 파일이 포함되어있지 않으므로 다음의 파일을 S08P22A308/mongttang-wallet-back/config/index.js 에 세팅해야 빌드가 가능하다.

```
/**
 * TODO: 개발 및 배포 환경 에 맞추어 아래의 상수들을 지정합니다.
 */
const API_BASE_URL = "통신할 Spring Server의 주소";
const BLOCKCHAIN_URL = "사용하는 Ethereum의 URL";
const BLOCKCHAIN_WEBSOCKET_URL = "사용하는 Ethereum의 Web Socket URL";
const NFT_CONTRACT_ADDRESS = "배포된 nft contract의 주소"
const MTT_CONTRACT_ADDRESS = "배포된 erc20 토큰의 주소";
const SSF_CONTRACT_ADDRESS = "교환에 사용할 사용화된 erc20 토큰의 주소";
const OWNER_PRIVATE_KEY =
  "contract 배포자의 privateKey";
const DECRYPTION_KEY = "상기 Spring Server에서 설정한 값";
const DECRYPTION_IV = "상기 Spring Server에서 설정한 값";

export {
  API_BASE_URL,
  BLOCKCHAIN_URL,
  BLOCKCHAIN_WEBSOCKET_URL,
  NFT_CONTRACT_ADDRESS,
  MTT_CONTRACT_ADDRESS,
  SSF_CONTRACT_ADDRESS,
  OWNER_PRIVATE_KEY,
  DECRYPTION_KEY,
  DECRYPTION_IV,
};
```

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/mongttang-wallet-back

혹은 clone 받은 S08P22A308/mongttang-wallet-back에

Dockerfile 생성한다.

Dockerfile

```
FROM node as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent --production
```

```
# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app

EXPOSE 4000
CMD [ "node", "app.js"]
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/mongttang_wallet_back/mongttang-wallet-back
docker build -t mongttang-wallet-back .
if [ $( docker ps -a | grep mongttang-wallet-back | wc -l ) -gt 0 ]; then
echo "mongttang-wallet-back container exists"
echo "delete container"
docker stop mongttang-wallet-back
docker rm mongttang-wallet-back
else
echo "mongttang-wallet-back container does not exist"
fi
docker run -p 4000:4000 --name mongttang-wallet-back -d mongttang-wallet-back
```

FRONTEND(지갑)

주의사항

gitlab에는 index.js 파일이 포함되어있지 않으므로 다음의 파일을 S08P22A308/mongttang-wallet-front/src/config/index.js에 세팅해야 빌드가 가능하다.

```
/**
 * TODO: 개발 및 배포 환경 에 맞추어 아래의 상수들을 지정합니다.
 */
const API_BASE_URL = "통신할 Spring Server의 주소";

export {
  API_BASE_URL,
};
```

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/mongttang-wallet-front

혹은 clone 받은 S08P22A308/mongttang-wallet-front에

Dockerfile과 default.conf를 생성한다.

Dockerfile

```
FROM node as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent --production
RUN npm install @vue/cli-service

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build
FROM nginx:1.13.9-alpine

# nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm -rf /etc/nginx/conf.d
```

```
COPY default.conf /etc/nginx/conf.d/default.conf

# 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/dist /usr/share/nginx/html

# 80포트 오픈하고 nginx 실행
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

default.conf

```
server {
  listen 80;
  location / {
    root   /usr/share/nginx/html;
    index  index.html;
    try_files $uri $uri/ /index.html;
  }
}
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/mongttang_wallet_front/mongttang-wallet-front
docker build -t mongttang-wallet-front .
if [ $( docker ps -a | grep mongttang-wallet-front | wc -l ) -gt 0 ]; then
echo "mongttang-wallet-front container exists"
echo "delete container"
docker stop mongttang-wallet-front
docker rm mongttang-wallet-front
else
echo "mongttang-wallet-front container does not exist"
fi
docker run -p 3333:80 --name mongttang-wallet-front -d mongttang-wallet-front
```

데이터베이스

MariaDB

Spring Data JPA를 사용하였으므로, spring.jpa.hibernate.ddl-auto=create 사용 시 자동으로 데이터베이스가 생성된다.

ec2 서버에서 maria db를 설치하고 사용하기 위해서는 아래의 방법을 따른다.

```
- mariadb 설치하기
  - 먼저 timezone을 확인하고 변경한다.
    - 확인 : -date
    - 변경 : sudo timedatectl set-timezone 'Asia/Seoul'

  - 설치하기
    - 패키지 목록 최신화 : sudo apt update
    - 서버 설치 : sudo apt install mariadb-server
    - 마리아 db가 확실히 실행되는지 다시 체크 : sudo systemctl start mariadb.service  ==(동일)  sudo service mariadb start

  - 보안 설정
    - sudo mysql_secure_installation
    - root 사용자의 password는 설정하지 않기 때문에
      enter current password for root : 에서는 enter
      set root password? 에서 n을 입력해준다.
    - 익명 유저, test 데이터베이스를 없애고, root 원격 로그인을 막기 위해 Y를 입력해준다.
```

```
- 새 유저 설정
  # 중요
  - MariaDB 10.3부터는 루트유저는 unix_socket을 사용하기 때문에 비밀번호를 사용하지 않는다. 문제가 생기는 것을 막기 위해 건드리지 말도록 하자.
  - sudo mysql로 root유저 접속이 가능하다.
  - 유저 생성 및 권한 주기
    ```
    USE mysql;
    //새로 유저를 만든다. 내부 접속용이면 localhost를, 외부에서 접근을 모두 허용해주려면 %를 입력해준다.
    CREATE USER '유저이름'@'접속ip' IDENTIIFED BY '비밀번호';
    //유저에게 권한을 부여해준다.
    GRANT ALL PRIVILEGES ON 데이터베이스이름.* TO '유저이름'@'접속ip';
    //설정을 저장한다.
    FLUSH PRIVILEGES;
    ```
  - unix_socket 사용을 원하지 않는다면 plugin을 공란으로 써주면 된다.
    ```
    USE mysql;
    UPDATE user SET plugin='' where user='root';
    SET password = password('비밀번호');
    FLUSH PRIVILEGES;
    ```
  - 유저 정보 변경은 update문을 이용하면 된다. flush privileges만 잊지말자
    ex) update user set password = password('새비밀번호') where user = '아이디' and host = '호스트 ip';
    flush privileges;

- 외부에서 접근을 원한다면
  - sudo vi /etc/mysql/mariadb.conf.d/50-server.cnf
  - sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf (vim 설치시)
  - 주석처리되어있는 port = 3306을 주석 해제해준다.
  - bind-address를 0.0.0.0으로 변경해준다.
```

BlockChain(Ethereum)

Solidity(Smart Contract) 배포 방법

S08P22A308/smart-contracts/contracts의 Mongttang.sol을 먼저 원하는 이더리움 네트워크에 배포한다. 생성자에 필요한 parameter는 이름(string), 심볼(string), 총 공급할 양(uint256)이며, 총 배포량은 즉시 배포자의 지갑에 들어오게 된다.

그 후, 배포된 contract의 address를 parameter로 사용하여 MongttangNFT.sol을 배포한다. Mongttang.sol과 같은 계정으로 진행해야 함에 주의한다.

이후,