포팅매뉴얼

1. 버전

Frontend

- node : 14.19.2
- npm : 6.14.17
- next : 12.3.1
- next-redux-wrapper : 8.0.0
- react : 18.2.0
- redux : 8.0.4

Backend

- JDK : 11
- Spring boot : 2.6.12
- JPA : 2.6.13
- Spring Cloud gateway : 3.1.4
- Spring Cloud eureka : 3.1.4
- Spring Cloud config : 3.1.5
- mySql : 8.0.29
- redis : 5.0.7
- AWS S3 : 2.2.6

Blockchain

- Geth : 1.10.26
- Go : go1.18.5
- solidity : 0.8.0
- web3 : 1.8.0

2. 배포

8080포트 → jenkins

80,443포트 → nignx

3000포트 → front

8000포트 → back

(1) Frontend 자동 배포

▼ 젠킨스 front pipeline과 Gitlab webhook을 통해 연결

▼ front pipeline 스크립트 설정

frontend 브랜치의 front 폴더를 도커에 build하고 3000포트로 도커 run

```
pipeline {
    agent any
    tools {
        nodejs "node14"
        git "Default"
    }
    stages {
        stage('prepare') {
            steps {
                echo 'prepare'
                 git branch: "frontend", credentialsId: "haengsong", url: 'https://lab.ssafy.com/s07-final/S07P31C103.git'
                 sh  'ls -al'
            }
        }
        stage('build') {
            steps {
                dir("front") {
                    sh 'ls -al'
                    sh 'if (sudo docker ps | grep "next"); then sudo docker stop next; fi'
                    sh 'sudo docker build -t next .'
                }


            }
        }
        stage('deploy') {
            steps {
                sh "ls -al"
                echo 'deploy'
                sh 'sudo docker run -it -d --rm -p 3000:3000 --name next next'
            }
        }
    }
}
```

▼ dockerfile 설정

```
FROM node:14-alpine

WORKDIR /usr/src/next

RUN npm i --global pm2

COPY ./package*.json ./

RUN npm i

COPY ./ ./

RUN npm run build

EXPOSE 3000

CMD ["npm", "run", "start" ]
```

(2) Backed 자동 배포

▼ 젠킨스 back pipeline과 Gitlab webhook을 통해 연결

▼ back pipeline 스크립트 설정

```
pipeline {
    agent any
    tools {
        nodejs "node14"
        git "Default"
    }
    stages {
        stage('prepare') {
            steps {
```

```
                echo 'prepare'
                 git branch: "spring", credentialsId: "haengsong", url: 'https://lab.ssafy.com/s07-final/S07P31C103.git'
                 sh  'ls -al'
            }
        }
        stage('build') {
            steps {
                dir("back/spring") {
                    sh 'ls -al'
                    sh 'if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi'
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build'
                     sh 'sudo docker build -t spring .'
                }


            }
        }
        stage('deploy') {
            steps {
                sh "ls -al"
                echo 'deploy'
                sh 'sudo docker run -it -d --rm -p 8000:8000 --name spring spring'
            }
        }
    }
}
```

▼ dockerfile 설정

```
FROM openjdk:11-jdk

ARG JAR_FILE=./build/libs/dog-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]
```

(3) SSL

▼ 가비아에서 도메인 설정

```
dog-hoogam.site
```

▼ Let's Encrypt SSL 인증서 발급

1. certbot 설치
2.

```
$ certbot certonly -d dog-hoogam.site --manual --preferred-challenges dns
```

입력후 yes or no 뜰 경우 Y 엔터

```
Please deploy a DNS TXT record under the name
_acme-challenge.new.lesstif.com with the following value:

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Before continuing, verify the record is deployed.
```

위에 나온 코드를 복사해서 가비아 DNS설정에서 호스트를 _acme-challenge으로

위치/값에 위의 코드를 입력후 우분투 창에서 엔터

성공적으로 ssl pem키 발급

▼ nginx 설정

/etc/nginx/sites-available/default 파일을 다음과 같이 추가

```
.
.
.
server {
  listen 80 default_server;
        listen [::]:80 default_server;
        server_name dog-hoogam.site;
        location /static {
                alias /home/ubuntu;
        }

        return 301 https://dog-hoogam.site$request_uri;
}

server{


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

  listen [::]:443 ssl ipv6only=on;
  listen 443 ssl;
  ssl_certificate /etc/letsencrypt/live/dog-hoogam.site/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/dog-hoogam.site/privkey.pem;
  include /etc/letsencrypt/options-ssl-nginx.conf;
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

  location / {
    proxy_pass http://k7c103.p.ssafy.io:3000;
    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;
    proxy_set_header X-Forwarded-Proto $scheme;
  }
}
.
.
.
```

(4) BlockChain 배포

▼ geth 설치

```
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

▼ genesis.json 생성하기

- Geth를 이용해 Ethereum private Network를 구성하기 위해서는 제네시스 블록을 생성해야한다.
- 만들 폴더 생성

  `mkdir -p dev/eth_localdata`
- CLI에서 편집을 위해 nano 편집기 사용

  `$ nano genesis.json // nano 생성할 파일이름`
- 명세에 맞춰서 생성 (공식문서 참고 : https://geth.ethereum.org/docs/interface/private-network)

```
{
  "config": {                   // 구성하고자 하는 이더리움 Network 설정
    "chainId": 921,
    "homesteadBlock": 0,
    "eip150Block": 0,
```

```
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "ethash": {}
  },
  "nonce": "0xdeadbeefdeadbeef",
  "difficulty": "0x10",
  "gasLimit": "9999999",
  "alloc": {}
}
```

▼ geth 초기화(init)

```
~/blockchain/dev/eth_localdata$ geth --datadir ~/blockchain/dev/eth_localdata/ init ~/blockchain/dev/eth_localdata/genesis.json
```

▼ geth 실행

```
nohup geth --networkid 921 --maxpeers 2 --datadir ~/blockchain/dev/eth_localdata/ --port 30303 --allow-insecure-unlock --http --ht
```

geth attach http://3.34.135.147:8545 (외부에서 접속 할 때)

3. 외부서비스키

```
// 블록체인 관련 키
NEXT_PUBLIC_PINATA_API_KEY=f1ff1732e9462610ced2
NEXT_PUBLIC_PINATA_API_SECRET=28c83000cfa7d7e05a9ac5d9b4561231d4a6486eec5a0547a61b8fbc41ea9d5e
NEXT_PUBLIC_COINBASE=0xC5Ec3ED8044a4ec50333CbBb096a17d5FD15372D
NEXT_PUBLIC_COINBASE_PASSWORD=c!103@bc#
NEXT_PUBLIC_GETH_NODE=http://3.34.135.147:8545
// 카카오 관련 키
NEXT_PUBLIC_KAKAO_KEY=f9faeb79aa8b66b75abbaf984a615309
```

```
cloud:
  aws:
    credentials:
      access-key: AKIAT4XDCDFHJJ755RPL
      secret-key: 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

jwt:
  secret: dyAeHubOOc8KaOfYB6XEQoEj1QzRlVgtjNL8PYs1A1tymZvvqkcEU7L1imkKHeDa

oauth2:
      client:
        registration:
          kakao:
            client-id: 9acb3e1fec7d5dcfa3a8577c21d604da
            client-secret: UQmeiRtYD5EyWqsQwI91DV6dpCUfmD8n
```