C206 맛트로 서비스 포팅 매뉴얼

1. 개발 환경

   Java development kit (JDK) : 11.0.16

   Nginx : 1.18.0 (Ubuntu)

   Springboot : 2.7.3

   Gradle : 7.5

   Intellij : 2022.1.3

   MongoDB : 6.0.1

   Docker : 20.10.18

   AWS : Ubuntu 20.04.4 LTS

   Jenkins : 2.361.1

   Spark : 3.1.3 for prebuilt Hadoop 2.7

   Zeppelin : 0.10.1

   Node.js : 16 or 14.19.0

   Npm :8.11.0

   Next-Js: 12.2.5

   React.js: 18.2.0

   Sass : 1.54.8

2. 환경 변수 설정

   2-1 스파크 환경 변수 설정

   Spark 설치 시 생성되는 spark 폴더를 최상단 디렉토리에 /opt/spark로 이동시켜줌.

   .bashrc 하단에

   ```
   export SPARK_HOME=/opt/spark/
   export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
   export MASTER=spark://spark-master:7077
   ```

export ZEPPELIN_HOME=/home/ubuntu/zeppelin/

export PATH=$ZEPPELIN_HOME/bin:$PATH

위와 같이 작성.

2-2 Zeppelin 환경 변수 설정

Ubuntu에 Spark, JDK의 설치 확인(--version)

Zeppelin 설치 url을 통해 zeppelin-0.10.1-bin-all.tgz 다운로드 및 압축 해제

/home/ubunt/.bashrc 하단에

export ZEPPELIN_HOME=/home/ubuntu/zeppelin/

export PATH=$ZEPPELIN_HOME/bin:$PATH

위 환경변수 추가.

2-3 도커 컴포즈 파일

```
version: '3'
services:
  mongodb:
    image: mongo
    container_name: mongodb
    restart: always
    ports:
      - 27017:27017
    volumes:
      - /data/mongodb:/data/db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=1234
      - MONGO_INITDB_DATABASE=mydb
```

2-4 엔진엑스 설정

```
server {
   listen 80; #80포트로 받을 때
   server_name j7c206.p.ssafy.io; #도메인주소, 없을경우 localhost
   return 301 https://j7c206.p.ssafy.io$request_uri; #리다이렉트 https url로

}

server {
      listen [::]:443 ssl ipv6only=on; # managed by Certbot
      listen 443 ssl; # managed by Certbot
      ssl_certificate /etc/letsencrypt/live/j7c206.p.ssafy.io/fullchain.pem; # managed by Certbot
      ssl_certificate_key /etc/letsencrypt/live/j7c206.p.ssafy.io/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
      include /etc/nginx/conf.d/service-url.inc;

      location / {
            proxy_pass http://j7c206.p.ssafy.io:3000; # /로 들어올 시에 프론트 포트로 연결해주고 https 설정
            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;
        }

        location /node {
            rewrite ^/node(.*)$ $1 break;
            proxy_pass http://localhost:8000; # /로 들어올 시에 프론트 포트로 연결 해주고 https 설정
            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;

            #Websocket support
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }

        location /api {
            proxy_pass $service_url; # /api로 들어올 시에 백 포트로 연결 해주고 https 설정
            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;
        }
    }
```

3. 백엔드 빌드

3-1 JDK 설치

sudo apt-get update

sudo apt-get **install** openjdk-11-**jdk**

3-2 도커 설치

sudo apt-get install docker

3-3 몽고 디비 컨테이너 실행

위에 언급된 도커 컴포즈 파일 참조

3-4 젠킨스 컨테이너 실행

docker pull Jenkins

docker run -d -p 8085:8080 -v /var/jenkins:/var/jenkins_home -v
/var/run/docker.sock:/var/run/docker.sock --name jenkins -u root {젠킨스컨테이너이름}

3-5 엔진엑스 설치 및 실행

sudo apt-get update

sudo apt-get install certbot python3-certbot-nginx
sudo certbot --nginx -d j7c206.p.ssafy.io

4. 프론트엔드 빌드

4-1. NEXT Dockerfile

```
FROM node:alpine

ARG NEXT_PUBLIC_KAKAO_API_KEY

WORKDIR /usr/app

RUN npm i --global pm2

COPY ./package*.json ./

RUN npm i

COPY ./ ./

RUN npm run build
```

```
EXPOSE 3000

USER node

# CMD [ "npm", "start" ]
CMD [ "pm2-runtime", "npm", "--", "start" ]
```

4 – 2 프론트 엔진엑스 default.conf (설정 파일)

```
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=STATIC:10m
inactive=7d use_temp_path=off;

upstream nextjs_upstream {
  server nextjs:3000;
}

server {
  listen 80 default_server;

  server_name _;

  server_tokens off;

  gzip on;
  gzip_proxied any;
  gzip_comp_level 4;
  gzip_types text/css application/javascript image/svg+xml;

  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection 'upgrade';
  proxy_set_header Host $host;
  proxy_cache_bypass $http_upgrade;

  location /_next/static {
    proxy_cache STATIC;
    proxy_pass http://nextjs_upstream;

    # 캐시테스트
    # For testing cache - remove before deploying to production
    # add_header X-Cache-Status $upstream_cache_status;
  }

  location /static {
    proxy_cache STATIC;
```

```
    proxy_ignore_headers Cache-Control;
    proxy_cache_valid 60m;
    proxy_pass http://nextjs_upstream;

    # 캐시테스트
    # add_header X-Cache-Status $upstream_cache_status;
  }

  location / {
    proxy_pass http://nextjs_upstream;
  }
}
```

4 – 3 프론트 엔진엑스 Dockerfile

```
FROM nginx:alpine

RUN rm /etc/nginx/conf.d/*

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

EXPOSE 80

CMD [ "nginx", "-g", "daemon off;" ]
```

4 – 4 노드 Dockerfile

```
FROM node:alpine

WORKDIR /usr/app

COPY ./package*.json ./

RUN npm i

COPY ./ ./

EXPOSE 8000
```

```
USER node

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

4 – 5 전체 젠킨스 빌드 파일

```
pipeline {
    agent any
    stages {
        stage("build"){
            steps{
                script{
                    try {
                        sh 'docker stop mattro'
                        sh 'docker rm mattro'
                        sh 'docker rmi wlsgh97/mattro'
                    } catch (e) {
                        sh 'echo "nextjs stop 중 실패"'
                    }
                    try {
                        sh 'docker stop mattro-nginx'
                        sh 'docker rm mattro-nginx'
                        sh 'docker rmi wlsgh97/mattro-nginx'
                    } catch (e) {
                        sh 'echo "nginx stop 중 실패"'
                    }
                    try {
                        sh 'docker stop mattro-node'
                        sh 'docker rm mattro-node'
                        sh 'docker rmi wlsgh97/mattro-node'
                    } catch (e) {
                        sh 'echo "nginx stop 중 실패"'
                    }
                    try{
                        sh 'docker build -t wlsgh97/mattro ./frontend --
build-arg NEXT_PUBLIC_KAKAO_API_KEY=${NEXT_PUBLIC_KAKAO_API_KEY}'
                        sh 'docker build -t wlsgh97/mattro-
nginx ./frontend/nginx'
                        sh 'docker build -t wlsgh97/mattro-
node ./frontend/node'
                    }catch(e){
                        sh 'echo "docker 이미지빌드중 실패"'
                    }
                }
```

```
            }
            post {
                success{
                    sh 'echo build 성공'
                }
                failure{
                    sh 'echo build 실패'
                }
            }
        }
        stage("create network"){
            steps{
                script{
                    try {
                        sh 'docker network remove my-network'
                        sh 'docker network create my-network'
                    } catch (e) {
                        sh 'echo "네트워크 생성중 실패"'
                    }
                }
            }
            post {
                success{
                    sh 'echo create 성공'
                }
                failure{
                    sh 'echo create 실패'
                }
            }
        }
        stage("docker-image push"){
            steps{
                sh 'echo "$DOCKER_HUB_PASSWORD" | docker login -u
"$DOCKER_HUB_ID" --password-stdin'
                sh "docker push wlsgh97/mattro"
                sh "docker push wlsgh97/mattro-nginx"
                sh "docker push wlsgh97/mattro-node"
            }
            post {
                success{
                    sh 'echo push 성공'
                }
                failure{
                    sh 'echo push 실패'
                }
            }
        }
        stage("run"){
```

```
            steps {
                sh 'docker run -d --network my-network --name mattro
wlsgh97/mattro'
                sh 'docker run -d -p 3000:80 --network my-network --link
mattro:nextjs --name mattro-nginx wlsgh97/mattro-nginx'
                sh 'docker run -d -p 8000:8000 --network my-network --name
mattro-node wlsgh97/mattro-node'
            }
        }
    }
}
```

5. 외부 서비스

5-1. 카카오톡 공유하기 기능

카카오 개발자에 앱 등록 후 키 발급 받은 뒤 서비스 정보 입력.

```
const { Kakao, location } = window;

// 공유하기 기능을 위해 initialize 마운트 될때 적용
if (!window.Kakao.isInitialized()) {
  window.Kakao.init(process.env.NEXT_PUBLIC_KAKAO_API_KEY);
}
Kakao.Link.sendDefault({
  objectType: "feed",
  content: {
    title: name,
    description: address,
    imageUrl: mainImageURL !== null ? mainImageURL : menuImageUrl,
    link: {
      mobileWebUrl: `https://j7c206.p.ssafy.io/${url}`,
      webUrl: `https://j7c206.p.ssafy.io/${url}`
    }
  }
});
};
```