포팅메뉴얼

INFRA

MySQL

1. 설치

```
sudo apt-get update
sudo apt-get install mysql-server
```

2. 접속 설정

```
sudo ufw allow mysql
```

3. 설정파일 변경

```
# cd /etc/mysql/mysqld.conf.d
# sudo vim mysqld.cnf

# bind-address 변경
bind-address = 0.0.0.0
```

Nginx, certbot

1. 설치

```
# nginx 설치
sudo apt-get update
sudo apt install nginx

sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# 이후 certbot 설치
sudo apt-get update
sudo apt-get install python3-certbot-nginx
```

2. 설정

```
# certbot
# 설치된 certvot을 이용하여 도메인(example.com)에 대한 SSL 인증서 발급
sudo certbot certonly --nginx -d [도메인]

# 다음 경로에 5개의 파일(4개의 .pem, 1개의 readme) 생성 확인
sudo ls -al /etc/letsencrypt/live/[도메인]


# 90일마다 만료되는 인증서 자동 갱신
sudo certbot renew --dry-run

# nginx
# # cd /etc/nginx/sites-enabled
# # sudo vim default

# /etc/nginx/sites-en
server {
  listen 80; #80포트로 받을 때
  server_name [도메인주소]; #도메인주소, 없을경우 localhost
  return 301 [도메인주소]$request_uri;

}
server {
  listen 443 ssl;
  server_name [도메인주소] www.[도메인주소];
```

```
  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/j8b309.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/j8b309.p.ssafy.io/privkey.pem;

  location / { # 프론트엔드
    proxy_pass http://localhost:[프론트엔드포트];
  }

  location /api { # 백엔드
    proxy_pass http://localhost:[백엔드포트];
    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; # https 필요
  }
}
```

3. nginx 재시작

```
sudo service nginx restart
```

Docker

1. 설치

```
sudo apt-get update

# 도커에 필요한 것 설치
# #- apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근 가능하게 함
# #- ca-certificate : certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM파일이 포함되어 있어 SSL 기반 앱이 SSL연결이 되어있는지 확인 가
# #- curl : 특정 웹사이트에서 데이터를 다운로드 받을 때 사용
# #- soft-ware-properties-common : *PPA를 추가하거나 제거할 때 사용
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common

# 도커 설치
sudo apt-get install docker-ce
```

Jenkins

1. 설치

```
sudo docker run -d \
-u root \
-p 9090:8080 \
--name=jenkins \
-v /home/ubuntu/docker/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-v "$HOME":/home/jenkinsci/blueocean \
jenkinsci/blueocean
```

2. jenkins 컨테이넌 접속해서 password 가져오기

```
sudo docker exec -it [컨테이너 이름] bash
# container 안쪽
cat /var/jenkins_home/secrets/initialAdminPassword
```

3. jenkins 업데이트
4. dockerfile / jenkinsfile

   java 백엔드 dockerfile

```
# openjdk:11 을 설치(빌더버전으로)
FROM openjdk:11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod =x ./gradlew
RUN ./gradlew bootJar

FROM openjdk:11
COPY --from=builder build/libs/newnews-0.0.1-SNAPSHOT.jar newnews.jar

EXPOSE 8080

CMD ["java","-jar","newnews.jar"]
```

java 백엔드 jenkinsfile

```
pipeline{
    agent any
    environment {
       BACK_CONTAINER_NAME="newnews_back_container"
       BACK_NAME = "newnews_back"
    }
    stages {
        stage('Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${BACK_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${BACK_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }
        stage('Build') {
            steps {
                script{
                    sh "sed -i 's/\${DB_USERNAME}/${DB_USERNAME}/' '${WORKSPACE}/specialization/src/main/resources/application.yml
                    sh "sed -i 's/\${DB_PASSWORD}/${DB_PASSWORD}/' '${WORKSPACE}/specialization/src/main/resources/application.yml
                    sh "sed -i 's/\${DB_PORT}/${DB_PORT}/' '${WORKSPACE}/specialization/src/main/resources/application.yml'"
                    sh "sed -i 's/\${DB_DOMAIN}/${DB_DOMAIN}/' '${WORKSPACE}/specialization/src/main/resources/application.yml'"
                    sh "docker build -t ${BACK_NAME} ./specialization/."
                }
            }
        }
        stage('Deploy'){
            steps {
                sh "docker run -d --name=${BACK_CONTAINER_NAME} -p 8888:8080 ${BACK_NAME}"
                sh "docker image prune --force"
            }
        }
    }
}
```

frontend dockerfile

```
FROM node:18.15.0 as builder

RUN mkdir /app

WORKDIR /app

COPY . /app

RUN npm install

RUN npm install -g typescript

RUN npm run build

FROM nginx

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

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

```
COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80

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

frontend jenkinsfile

```
pipeline {
    agent any
    environment {
       FRONT_CONTAINER_NAME="newnews_front_container"
       FRONT_NAME = "newnews_front"
    }
    stages {
        stage('Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${FRONT_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${FRONT_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }
        stage('Build') {
            steps {
                script{
                    sh "docker build -t ${FRONT_NAME} ./newnews/."
                }
            }
        }
        stage('Docker run') {
            steps {
                sh "docker run -d --name=${FRONT_CONTAINER_NAME} -p 3000:80 ${FRONT_NAME}"
                sh "docker image prune --force"
            }
        }
    }
}
```

fastapi dockerfile

```
FROM ubuntu:latest

ENV LANG=C.UTF-8
RUN apt-get update
RUN apt-get install -y --no-install-recommends tzdata g++ curl

# install java
RUN apt-get update
RUN apt-get install -y openjdk-11-jdk
ENV JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"

# install python
RUN apt-get install -y python3-pip python3-dev
RUN cd /usr/local/bin && \
  ln -s /usr/bin/python3 python && \
  ln -s /usr/bin/pip3 pip && \
  pip install --upgrade pip

# apt clean
RUN apt-get clean && \
  rm -rf /var/lib/apt/lists/*

# copy resources
COPY . .

# install python package
RUN pip install -r requirements.txt

EXPOSE 8000

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

fastapi jenkinsfile

```
pipeline{
    agent any
    environment {
       BACK_CONTAINER_NAME="newnews_fastapi_container"
       BACK_NAME = "newnews_fastapi"
    }
    stages {
        stage('Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${BACK_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${BACK_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }
        stage('Build') {
            steps {
                script{
                    sh "sed -i 's/\${DB_DOMAIN_1}/${DB_DOMAIN_1}/' '${WORKSPACE}/fastapi/main.py'"
                    sh "sed -i 's/\${DB_DOMAIN_2}/${DB_DOMAIN_2}/' '${WORKSPACE}/fastapi/main.py'"
                    sh "sed -i 's/\${DB_DOMAIN_3}/${DB_DOMAIN_3}/' '${WORKSPACE}/fastapi/main.py'"
                    sh "sed -i 's/\${DB_DOMAIN_4}/${DB_DOMAIN_4}/' '${WORKSPACE}/fastapi/main.py'"
                    sh "sed -i 's/\${DB_PORT}/${DB_PORT}/' '${WORKSPACE}/fastapi/main.py'"
                    sh "docker build -t ${BACK_NAME} ./fastapi/."
                }
            }
        }
        stage('Deploy'){
            steps {
                sh "docker run -d --name=${BACK_CONTAINER_NAME} -p 8889:8000 ${BACK_NAME}"
                sh "docker image prune --force"
            }
        }
    }
}
```

DATA

EC2 자바 설치

```
# 업데이트
sudo apt-get update

# jdk 11 설치
sudo apt-get install openjdk-11-jdk

# 버전 확인
java -version
```

HADOOP & YARN 설치

```
# 하둡 tar파일 설치
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz

# 압축 풀기
tar -zxvf {hadoop 알집 파일 이름}
```

SPARK 설치

```
# 하둡 tar파일 설치
wget https://dlcdn.apache.org/spark/spark-3.3.2/spark-3.3.2-bin-hadoop3.tgz

# 압축 풀기
tar -zxvf {spark 알집 파일 이름}
```

ZOOKEEPER설치

```
# 하둡 tar파일 설치
wget https://dlcdn.apache.org/zookeeper/zookeeper-3.7.1/apache-zookeeper-3.7.1-bin.tar.gz

# 압축 풀기
tar -zxvf {zookeeper 알집 파일 이름}
```

KAFKA설치

```
# 하둡 tar파일 설치
wget https://downloads.apache.org/kafka/3.4.0/kafka_2.12-3.4.0.tgz

# 압축 풀기
tar -zxvf {kafka 알집 파일 이름}
```