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-enabled
server {
  listen 80; #80포트로 받을 때
  server_name k8b308.p.ssafy.io; #도메인주소, 없을경우 localhost
  client_max_body_size 100M;
  return 301 https://k8b308.p.ssafy.io$request_uri;
}

server {
  listen 443 ssl;
```

```
  server_name k8b308.p.ssafy.io www.k8b308.p.ssafy.io;
  client_max_body_size 100M;
  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/k8b308.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/k8b308.p.ssafy.io/privkey.pem;

        location /api { # 백엔드
                client_max_body_size 100M;
                proxy_pass http://localhost:1124/api;
                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
        }
        location /swagger-ui {
                proxy_pass http://localhost:1124/swagger-ui;
                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
        }
        location /topics {
                proxy_pass http://localhost:8082/topics;
                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/another_back-0.0.1-SNAPSHOT.jar another.jar

EXPOSE 8080

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

java 백엔드 jenkinsfile

```
pipeline{
    agent any
    environment {
       BACK_CONTAINER_NAME="another_back_container"
       BACK_NAME = "another_back"
    }
    stages {
        stage('Create Directory') {
            steps {
                sh 'mkdir -p /home/ec2-user'
            }
        }
        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}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${DB_PASSWORD}/${DB_PASSWORD}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${DB_PORT}/${DB_PORT}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${DB_DOMAIN}/${DB_DOMAIN}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${SECRET_KEY}/${SECRET_KEY}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${ACCESS_KEY}/${ACCESS_KEY}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${HDFS_URL}/${HDFS_URL}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "sed -i 's/\${HDFS_PORT}/${HDFS_PORT}/' '${WORKSPACE}/another_back/src/main/resources/application.yml'"
                    sh "docker build -t ${BACK_NAME} ./another_back/."
                }
            }
        }
        stage('Deploy'){
            steps {
                sh "docker run -d --name=${BACK_CONTAINER_NAME} -v /home/ec2-user:/home/ec2-user -p 1124:8080 ${BACK_NAME}"
                sh "docker image prune --force"
            }
        }
    }
}
```

DATA jenkinsfile

```
pipeline{
    agent any
    environment {
       DATA_CONTAINER_NAME="pipeline-confluent-kafka-rest"
       DATA_NAME = "pipeline-confluent-kafka-rest"
    }
    stages {
        stage('Clean'){
            steps{
                script {
                    try{
                        sh "docker stop ${DATA_CONTAINER_NAME}"
                        sleep 1
                        sh "docker rm ${DATA_CONTAINER_NAME}"
                    }catch(e){
                        sh 'exit 0'
                    }
                }
            }
        }
        stage('Build') {
            steps {
                script{
                    sh "sed -i 's/\${DATA_SERVER}/${DATA_SERVER}/' '${WORKSPACE}/another_data/confluent_config/kafka-rest.properti
                    sh "docker build -t pipeline-confluent-kafka-rest ./another_data/."
                }
            }
        }
        stage('Deploy'){
            steps {
                sh "docker run -d --name=${DATA_CONTAINER_NAME} -p 8082:8082 ${DATA_NAME}"
                sh "docker image prune --force"
            }
        }
    }
}
```

DATA_dockerfile

```
FROM ubuntu:20.04
RUN mkdir -p /root/install
RUN apt-get update

WORKDIR /root/install

ENV DEBIAN_FRONTEND noninteractive
ENV JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

RUN apt-get install openjdk-11-jdk -y
RUN apt-get install wget -y
RUN apt-get install vim -y

# confluent-community 설치
RUN wget http://packages.confluent.io/archive/7.3/confluent-community-7.3.3.tar.gz
RUN tar -zxvf confluent-community-7.3.3.tar.gz
RUN mv confluent-7.3.3 /usr/local/confluent

# kafka-rest 설정파일 복사
COPY confluent_config/kafka-rest.properties /usr/local/confluent/etc/kafka-rest/kafka-rest.properties
RUN sed -i 's/\r//g' /usr/local/confluent/etc/kafka-rest/kafka-rest.properties

# kakfa-rest 실행
CMD /usr/local/confluent/bin/kafka-rest-start /usr/local/confluent/etc/kafka-rest/kafka-rest.properties
```

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-3.4.0/spark-3.4.0-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}
```

SPARK CODE

```
screen -r endZero
spark-submit --master local[*] --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 endZero.jar

screen -r endOne
spark-submit --master local[*] --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 endOne.jar

screen -r endTwo
spark-submit --master local[*] --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0 endTwo.jar

screen -r kafka
zkServer.sh start ./zookeeper/conf/another.cfg
kafka-server-start.sh ~/kafka/config/server.properties
```