포팅메뉴얼

Simollu

A701 - 12시 6분 조

팀원 : 이소민 고태진 김동언 김수형 이민정 임혜은

기술

FrontEnd

- Flutter
- 지도 APP
  - TMAP
  - Kakao Map
  - Google Map
- FCM

BackEnd

- Java 11.0.17
- SpringBoot 2.7.11
- JPA
- OAuth2
- Spring Security
-

구성

2개의 서버로 구성되어 있고 각각 스프링부트를 모아놓은 서비스 서버와 DB 서버로 사용합니다.

서비스 서버

- Config
- Eureka
- Spring Cloud Gateway

- User
- Restaurant
- Waiting
- Alert
- Calculator
- Nginx
- Jenkins

DB 서버

- MySQL
- Redis
- Elastic Search

Docker 설치

아래의 명령어를 순차적으로 입력합니다.

```
apt-get update
apt-get install sudo
apt-get install vim
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Docker Network 추가

도커 컨테이너를 하나의 네트워크로 관리해 줍니다.

```
sudo docker network create <your Docker Network Name>
```

방화벽 설정

22, 80, 443 포트를 ufw를 이용해 개방하고 실행합니다.

```
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

```
sudo ufw allow 8080/tcp
sudo ufw enable
```

Nginx 설치

아래의 명령어를 통해 nginx를 설치합니다.

```
sudo apt install nginx
```

폴더를 만들어 리버스 프록시 관련 설정 파일을 만듭니다.

```
cd/etc/nginx/conf.d
```

저는 두 개의 conf 파일을 사용 중 입니다.

- default.conf

```
upstream backend{
    server 127.0.0.1:8000;
}

server {
  server_name simollu.com;

  location /api {
        rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        proxy_set_header Host $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;
    }




    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/simollu.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/simollu.com/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

}


server {
    if ($host = simollu.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
  listen 80;
  server_name simollu.com;
    return 404; # managed by Certbot


}
```

- jenkins.conf

```
upstream jenkins {
  keepalive 32; # keepalive connections
  server 127.0.0.1:9090; # jenkins ip and port
}

# Required for Jenkins websocket agents
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server {       # Listen on port 80 for IPv4 requests

  server_name     jenkins.example.com;  # replace your server domain name

  # this is the jenkins web root directory
  # (mentioned in the output of "systemctl cat jenkins")
  root            /var/run/jenkins/war/;

  access_log      /var/log/nginx/jenkins.access.log;
  error_log       /var/log/nginx/jenkins.error.log;

  # pass through headers from Jenkins that Nginx considers invalid
  ignore_invalid_headers off;

  location ~ "^/static/[0-9a-fA-F]{8}\/(.*)$" {
    # rewrite all static files into requests to the root
    # E.g /static/12345678/css/something.css will become /css/something.css
    rewrite "^/static/[0-9a-fA-F]{8}\/(.*)" /$1 last;
  }

  location /userContent {
    # have nginx handle all the static requests to userContent folder
    # note : This is the $JENKINS_HOME dir
    root /var/lib/jenkins/;
    if (!-f $request_filename){
      # this file does not exist, might be a directory or a /**view** url
      rewrite (.*) /$1 last;
      break;
    }
    sendfile on;
  }

  location / {
      sendfile off;
      proxy_pass         http://jenkins;
      proxy_redirect     default;
      proxy_http_version 1.1;

      # Required for Jenkins websocket agents
      proxy_set_header   Connection        $connection_upgrade;
      proxy_set_header   Upgrade           $http_upgrade;

      proxy_set_header   Host              $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;
      proxy_max_temp_file_size 0;

      #this is the maximum upload size
      client_max_body_size       10m;
      client_body_buffer_size    128k;

      proxy_connect_timeout      90;
      proxy_send_timeout         90;
      proxy_read_timeout         90;
      proxy_buffering            off;
      proxy_request_buffering    off; # Required for HTTP CLI commands
      proxy_set_header Connection ""; # Clear for keepalive
  }
```

```
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/jenkins.simollu.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/jenkins.simollu.com/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

}


server {
    if ($host = jenkins.simollu.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen          80;

  server_name     jenkins.simollu.com;
    return 404; # managed by Certbot


}
```

SSL 적용

CertBot과 Let's Encrypt를 이용해 SSL을 적용합니다.

아래의 명령어로 certbot을 설치합니다.

```
sudo apt-get update
sudo snap install core; sudo snap refresh core
sudo snap install certbot --classic
```

SSL을 자동 적용합니다.

```
sudo certbot --nginx
```

질문에 대해 적절하게 대답을 하신 다음 SSL을 적용할 도메인을 선택합니다.

그 이후 nginx를 재시작 합니다.

```
sudo service nginx restart
```

Jenkins 설치

jenkins 설치

```
sudo docker pull jenkins/jenkins:lts-jdk11
```

jenkins 실행

```
docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v \
/var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:lts-jdk11
```

jenkins 컨테이너를 실행 후 로그를 확인해 기본 비밀번호를 찾습니다.

```
sudo docker logs jenkins
```

그 이후 관련 플러그인들을 설치합니다.

Jenkins 내부에서 Docker Build를 이용하기 위해 Jenkins 내부에 Docker를 설치합니다.

jenkins 내부 접근

```
sudo docker exec –it jenkins bash
```

jenkins 내부 docker 설치

```
apt-get remove docker docker-engine docker.io containerd runc
apt-get update
apt-get install ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

MySQL 설정

MySQL 이미지를 가져옵니다.

```
docker pull mysql:8.0.31
```

이미지를 실행 시킵니다.

```
docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=1234 --name mysql mysql:8.0.31
```

SpringBoot 실행

실제 프로젝트를 배포해야 할 때는 필요한 모든 파일들을 적절한 위치에 배치하고, 환경 변수와 API-KEY 등을 입력한 뒤 빌드를 진행해야 합니다.

Spring Cloud Config 를 통해 application.yml 파일들을 GitLab에 관리했습니다.

적절한 위치에 배치한 다음 빌드와 배포를 진행합니다.

- Config Service

```
mkdir -p /var/jenkins_home/workspace/config/ConfigService/src/main/resources
cp /var/jenkins_home/config/config-service/bootstrap.yml /var/jenkins_home/workspace/config/ConfigService/src/main/resources/
cd /var/jenkins_home/workspace/config/ConfigService
chmod +x ./gradlew
./gradlew clean build

cp /var/jenkins_home/config/config-service/Dockerfile /var/jenkins_home/workspace/config/ConfigService/
docker build -t configservice .
docker run -d -p 8888:8888 --network simollu-network --name config-service configservice
```

- Eureka

```
docker stop discovery-service || true
docker rm discovery-service || true
docker rmi discovery-service|| true


mkdir -p /var/jenkins_home/workspace/discovery/DiscoveryService/src/main/resources
cp /var/jenkins_home/config/discovery-service/bootstrap.yml /var/jenkins_home/workspace/discovery/DiscoveryService/src/main/resources/
cd /var/jenkins_home/workspace/discovery/DiscoveryService
chmod +x ./gradlew
./gradlew clean build

cp /var/jenkins_home/config/discovery-service/Dockerfile /var/jenkins_home/workspace/discovery/DiscoveryService/
docker build -t discovery-service .
docker run -d -p 8761:8761 --network simollu-network --name discovery-service discovery-service
```

- API Gateway

```
docker stop apigateway-service || true
docker rm apigateway-service || true
docker rmi apigateway-service|| true


mkdir -p /var/jenkins_home/workspace/apigateway/ApigatewayService/src/main/resources
cp /var/jenkins_home/config/apigateway-service/bootstrap.yml /var/jenkins_home/workspace/apigateway/ApigatewayService/src/main/resourc
cd /var/jenkins_home/workspace/apigateway/ApigatewayService
chmod +x ./gradlew
./gradlew clean build

cp /var/jenkins_home/config/apigateway-service/Dockerfile /var/jenkins_home/workspace/apigateway/ApigatewayService/
docker build -t apigateway-service .
docker run -d -p 8000:8000 --network simollu-network --name apigateway-service apigateway-service
```

- user Service

```
docker stop user-service || true
docker rm user-service || true
docker rmi user-service|| true


mkdir -p /var/jenkins_home/workspace/user/UserService/src/main/resources
cp /var/jenkins_home/config/user-service/bootstrap.yml /var/jenkins_home/workspace/user/UserService/src/main/resources/
cd /var/jenkins_home/workspace/user/UserService
chmod +x ./gradlew
./gradlew clean build

cp /var/jenkins_home/config/user-service/Dockerfile /var/jenkins_home/workspace/user/UserService/
docker build -t user-service .
docker run -d --network simollu-network --name user-service user-service
```

- restaurant service

```
docker stop restaurant-service || true
docker rm restaurant-service || true
docker rmi restaurant-service|| true


cp /var/jenkins_home/config/restaurant-service/bootstrap.yml /var/jenkins_home/workspace/restaurant/spring-data-elasticsearch/src/main
cd /var/jenkins_home/workspace/restaurant/spring-data-elasticsearch
chmod +x ./gradlew
./gradlew clean build

docker build -t restaurant-service .
docker run -d --network simollu-network --name restaurant-service restaurant-service
```

- waiting service

```
docker stop waiting-service || true
docker rm waiting-service || true
docker rmi waiting-service|| true

mkdir -p /var/jenkins_home/workspace/waiting/WaitingService/src/main/resources
cp /var/jenkins_home/config/waiting-service/bootstrap.yml /var/jenkins_home/workspace/waiting/WaitingService/src/main/resources/
cd /var/jenkins_home/workspace/waiting/WaitingService
chmod +x ./gradlew
./gradlew clean build

docker build -t waiting-service .
docker run -d --network simollu-network --name waiting-service waiting-service
```

- calculator service

```
docker stop calculator-service || true
docker rm calculator-service || true
docker rmi calculator-service|| true


mkdir -p /var/jenkins_home/workspace/calculator/CalculatorService/src/main/resources
cp /var/jenkins_home/config/calculator-service/bootstrap.yml /var/jenkins_home/workspace/calculator/CalculatorService/src/main/resourc
cd /var/jenkins_home/workspace/calculator/CalculatorService
chmod +x ./gradlew
./gradlew clean build

cp /var/jenkins_home/config/calculator-service/Dockerfile /var/jenkins_home/workspace/calculator/CalculatorService/
docker build -t calculator-service .
docker run -d --network simollu-network --name calculator-service calculator-service
```

- alert service

```
docker stop alert-service || true
docker rm alert-service || true
docker rmi alert-service|| true

mkdir -p /var/jenkins_home/workspace/alert/Alert/src/main/resources
mkdir -p /var/jenkins_home/workspace/alert/Alert/src/main/resources/firebase
cp /var/jenkins_home/config/alert-service/bootstrap.yml /var/jenkins_home/workspace/alert/Alert/src/main/resources/
cp /var/jenkins_home/config/alert-service/simollu_service_key.json /var/jenkins_home/workspace/alert/Alert/src/main/resources/firebase
cd /var/jenkins_home/workspace/alert/Alert
chmod +x ./gradlew
./gradlew clean build

docker build -t alert-service .
docker run -d --network simollu-network --name alert-service alert-service
```

위의 명령어를 순서대로 입력합니다.

Config Server 에서 설정 파일을 각각의 서비스에 연동해주고 Eureka에 등록되고 Api Gateway를 통해 경로 설정이 이루어 집니다.

설정파일의 저장소 url 은 다음과 같습니다.

https://lab.ssafy.com/cladren12332/simollu_config

현재는 보안상의 이유로 private으로 설정 되어있습니다.

요청을 하시면 권한을 부여해 접근하도록 하겠습니다.

ELK 설치 폴더 구조

```
config 폴더
    L jvm.options
logstash 폴더
    L config 폴더
        L logstash.yml
    L pipeline 폴더
        L logstash.conf
    L Dockerfile
Dockerfile
els.yml
jvm.options
```

jvm.options

```
# Elasticsearch의 메모리 할당량을 6G로 조정합니다.
-Xms3g
-Xmx3g
```

logstash.yml

```
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://es:9200" ]

path.plugins: ["/usr/share/logstash/plugins"]
```

logstash.conf

```
input {
  jdbc {
    jdbc_driver_library => "/usr/share/logstash/mysql-connector-java-8.0.26.jar"
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://**.**.***.***:3306/simollu"
    jdbc_user => "******"
    jdbc_password => "*******"
    jdbc_validate_connection => true
    statement => "SELECT * from search_terms"
    type => "mytype"
  }
}

output {
  elasticsearch {
    hosts => ["es:9200"]
    index => "myindex"
    document_type => "_doc"
  }
}
```

logstash Dockerfile

```
ARG ELK_VERSION

# https://www.docker.elastic.co/
FROM docker.elastic.co/logstash/logstash:7.15.2

ENV MYSQL_CONNECTOR_VERSION 8.0.26

# MySQL Connector/J 다운로드 및 설치
RUN curl -L -o /usr/share/logstash/mysql-connector-java-8.0.26.jar https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.26/my
  && chmod 644 /usr/share/logstash/mysql-connector-java-8.0.26.jar
```

Dockerfile

```
ARG VERSION
FROM docker.elastic.co/elasticsearch/elasticsearch:7.15.2
COPY ./config/jvm.options /usr/share/elasticsearch/config/
RUN elasticsearch-plugin install analysis-nori
```

els.yml

```
version: '3.7'
services:
  es:
    build:
      context: .
      args:
        VERSION: 7.15.2
    container_name: es
    environment:
      - node.name=single-node
      - cluster.name=backtony
      - discovery.type=single-node
    ports:
      - 8080:9200
    networks:
      - es-bridge

  logstash:
    build:
      context: logstash/
      args:
        ELK_VERSION: 7.15.2
    volumes:
      - type: bind
```

```
        source: ./logstash/config/logstash.yml
        target: /usr/share/logstash/config/logstash.yml
        read_only: true
      - type: bind
        source: ./logstash/pipeline
        target: /usr/share/logstash/pipeline
        read_only: true
      - type: bind
        source: ./logstash/input_data
        target: /usr/share/logstash/input_data
      - type: bind
        source: ./logstash/plugins
        target: /usr/share/logstash/plugins
        read_only: false
    ports:
      - 5555:5000
    environment:
      LS_JAVA_OPTS: "-Xmx256m -Xms256m"
    networks:
      - es-bridge
    depends_on:
      - es


  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:7.15.2
    environment:
      SERVER_NAME: kibana
      ELASTICSEARCH_HOSTS: http://es:9200
    ports:
      - 5601:5601
    # Elasticsearch Start Dependency
    depends_on:
      - es
    networks:
      - es-bridge

networks:
  es-bridge:
    driver: bridge
```