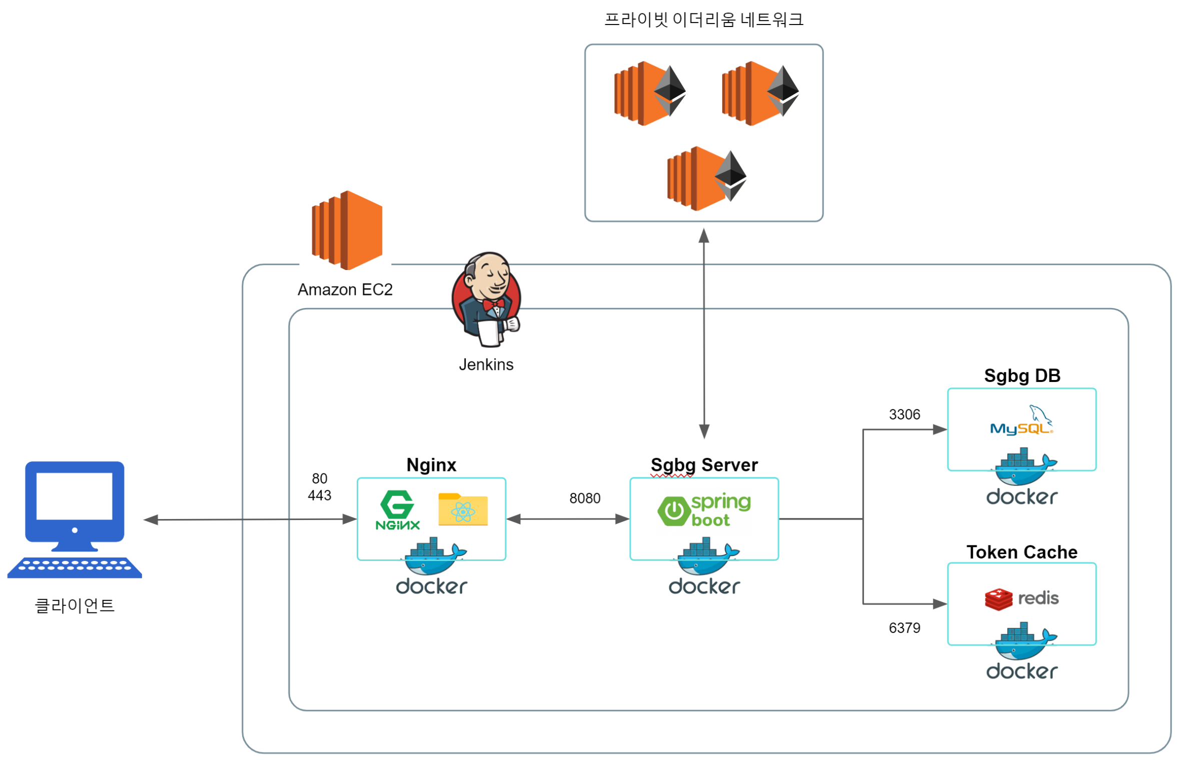
EC2에 배포하기

목차

들어가기 전, 아키텍처



Jenkins는 gitlab의 소스코드를 통해 도커라이징하는 등의 **CI/CD 파이프라인을 구축**하였고,

Docker-compose를 통해 그 외 nginx, mysql, redis, ganache-cli 등을 한 번에 컨테이너화 시키는 작업을 하였습니다.

1. Docker-compose 설정

Certbot + Nginx Container 로 HTTPS 적용

Nginx and Let's Encrypt with Docker in Less Than 5 Minutes

The other day, I wanted to quickly launch an nginx server with Let's Encrypt certificates. I expected the task to be easy and straightforward. Turns out: I was wrong, it took a significant amount of time and it's quite a bit more complicated. Of course, in the grand scheme of things, it is pretty straightforward.

https://pentacent.medium.com/nginx-and-lets-encrypt-with-docker-in-less-than-5-minutes-b4b8a60d3a71

HTTPS 인증서 검증 과정

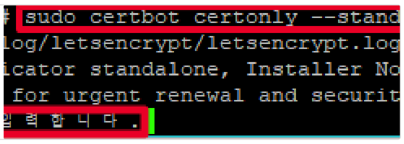
1. Let's Encrpyt가 도메인이 요청하면, 도메인 검증을 수행
2. Certbot이 검증 결과에 대한 응답 데이터를 제공
3. Nginx가 해당 응답 데이터를 전달(serve)해주는 것

⭐ Spring Boot SSL 설정

Spring Boot에 Let's Encrypt SSL 적용하기

아파치나 NginX 같은 웹서버 없이 Spring Boot으로 만든 웹 어플리케이션에 무료 SSL 중 하나인 Let's Encrypt을 적용하게 되었다. 90일동안 사용 가능하며 다시 갱신을 시켜줘야하나? 싫었지만 간단한 설정으로 자동갱신까지 가능하니 참 괜찮은 것 같다. 그리고 Let's Encrypt SSL 인증서 발급 방법은 여러가지가 있는데 그 중 standalone 방식을 택했다. 이제부터 standalone 방식으로 인증서를 발급받아 SSL을 적용해보자.

https://jiwontip.tistory.com/83



```
# sudo certbot certonly --stand
log/letsencrypt/letsencrypt.log
icator standalone, Installer No
 for urgent renewal and securit
입 력 합 니 다 .
```

```
sudo openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name ttp -CAfile chain.pem -caname root
```

pem은 스프링 부트에서 인식하지 못하므로, pem을 PKCS12형식으로 변경 후,

resources 폴더 아래에 위치시킨다.

배포용 `application.properties` 변경

로컬과 다른 부분 붉은 색 처리

```
# server.port=8080

# Key Store
server.ssl.enabled=true
server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=tldrmfqjdrmf

server.ssl.protocol=TLS
server.ssl.enabled-protocols=TLSv1.2

# BANNER
banner.location=classpath:banner.txt

# Charset of HTTP requests and responses. Added to the "Content-Type" header if not set explicitly.
server.servlet.encoding.charset=UTF-8
# Enable http encoding support.
server.servlet.encoding.enabled=true
# Force the encoding to the configured charset on HTTP requests and responses.
server.servlet.encoding.force=true

spring.devtools.livereload.enabled=true

# ===============================
# = DATA SOURCE
# ===============================
spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
spring.jpa.hibernate.naming.physical-strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
spring.data.web.pageable.one-indexed-parameters=true

spring.datasource.url=jdbc:mysql://j7a707.p.ssafy.io:3306/sgbg?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewri
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.username=root
spring.datasource.hikari.password={root_password}

# ===============================
# = LOGGING
# ===============================
logging.file.name=./sgbg-log.log
logging.level.root=INFO
logging.level.org.springframework.web=ERROR
logging.level.org.sringframework.boot=DEBUG
logging.level.org.apache.tiles=INFO
logging.level.com.sgbg=DEBUG


# Logging pattern for the console
logging.pattern.console=%d{HH:mm:ss} %clr(%5p) [%c] %m%n

# Logging pattern for file
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n

logging.file = logs/backend.log

# ===============================
# = Ethereum Client
# ===============================
spring.web3j.client-address=http://43.201.23.237:8545
spring.web3j.admin-client=true

# ===============================
# = Ethereum CA & EOA
# ===============================
# eth.encrypted.password=Pn0OlSN0SdhjNK5X2EhUPQ==
# eth.erc20.contract=0xcDcedAcea55DB472060042C644096DF65B9d9849
# eth.purchase.record.contract=0x0c2BC3cAB0036D1E51734B4D6e6cb8BCB046A2C7
# eth.admin.address=718ca8088ae9c120551defcbbecfedc3b9761a1f
# eth.admin.wallet.filename=admin.wallet

# Kakao REST API
kakao.client.id=b833a7474795b71dfb16b78fbcac80be
kakao.client.secret=ciUaftxmuIwrIWhekVk5c0mtH9H4KsNc
kakao.redirect.uri=https://j7a707.p.ssafy.io/login

# Spring doc
springdoc.swagger-ui.path=/index.html
springdoc.swagger-ui.groups-order=desc
springdoc.swagger-ui.operations-sorter=method
springdoc.swagger-ui.disable-swagger-default-url=true
springdoc.swagger-ui.display-request-duration=true
springdoc.api-docs.path=/api-docs
springdoc.default-consumes-media-type=application/json
springdoc.default-produces-media-type=application/json
springdoc.paths-to-match=/**

# Redis for token
spring.redis.host=j7a707.p.ssafy.io
#spring.redis.password=tldrmfqjdrmf
spring.redis.port=6379

# Cash Contract + Admin Address
eth.cash.contract=0x7eb1c5186f6daa1cd0959a3d9e01541bbd18af3c
eth.admin.address=0x0dbbb076130a1c48bc2511fbe313f02be02e1a57
eth.admin.private=0x922e800af24e78289a11218c60e0724817e01c9468a526c26d1141e9de4fe8c8
```

Docker-compose 작성

```
version: "3.7"
services:
    nginx:
        container_name: sgbg-nginx
        image: nginx:latest
        ports:
            - 80:80
            - 443:443
        volumes:
            - ./data/nginx/conf:/etc/nginx/conf.d
            - ./data/certbot/conf:/etc/letsencrypt
            - ./data/certbot/www:/var/www/certbot
        restart: always
    certbot:
        container_name: sgbg-certbot
        image: certbot/certbot
        volumes:
            - ./data/certbot/conf:/etc/letsencrypt
            - ./data/certbot/www:/var/www/certbot
    mysql:
        container_name: sgbg-mysql
        image: mysql:5.7
        ports:
            - 3306:3306
        volumes:
            - ./data/mysql:/var/lib/mysql
        environment:
            - MYSQL_ROOT_PASSWORD={ROOT_PASSWORD}
            - MYSQL_DATABASE=sgbg
        command:
            - --character-set-server=utf8
            - --collation-server=utf8_general_ci
        restart: always
    redis:
        container_name: sgbg-redis
        image: redis
        ports:
            - "6379:6379"
        volumes:
            - ./data/redis:/data
        restart: on-failure
```

`docker-compose up` 을 통해 한 번에 컨테이너화를 시킴

2. Nginx default.conf 설정

```
# geth or ganache 설정
server {
  listen 8545;
  server_name j7a707.p.ssafy.io;

  location / {
    proxy_pass http://172.26.10.156:8545;
  }
}

# 80번 포트 https로 redirect
server {
  listen 80;
  server_name j7a707.p.ssafy.io;
  location /.well-known/acme-challenge/ {
    root /var/www/certbot;
  }

  location / {
    return 301 https://j7a707.p.ssafy.io;
  }
}

server {
  listen 443 ssl;
  server_name j7a707.p.ssafy.io;

  ssl_certificate /etc/letsencrypt/live/j7a707.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/j7a707.p.ssafy.io/privkey.pem;

  # 보안을 위한 NGINX 설정
    include /etc/letsencrypt/options-ssl-nginx.conf; #certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; #certbot

  location / {
    root   /usr/share/nginx/html/build;
    index index.html index.htm;
    try_files $uri $uri/ /index.html;
  }

  location /api {
    rewrite ^/api(/.*)$ $1 break; # /api -> / 로 rewrite
    proxy_pass https://j7a707.p.ssafy.io:8080;
    proxy_connect_timeout 600;
        proxy_send_timeout 600;
        proxy_read_timeout 600;
        send_timeout 600;
    proxy_redirect off;

    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;
  }
}
```

3. Jenkins 설정

Backend 빌드 전, 사전 작성 Dockerfile

```
# 자바 버전
FROM openjdk:15-jdk-alpine as builder

# 현재 내 위치는 workspace
# workspace/SGBG/backend/src/main/resources/application.properties
WORKDIR /app

COPY ./SGBG/backend .

RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM openjdk:15-jdk-alpine
# war파일 복사
COPY --from=builder app/build/libs/*.jar ./app.jar
VOLUME ["/home/ubuntu/java-logs"]

# 포트번호 설정
EXPOSE 8080

# ENTRYPOINT 명령을 지정, app.war 실행
ENTRYPOINT ["java","-jar","/app.jar","--logging.level.org.springframework.web=ERROR", "--logging.level.org.sringframework.boot=DEBUG", "--logging.level.org.apache.tiles
```

Build Step 1: Backend 빌드

```
# deployback 컨테이너가 있으면 삭제 후 다시 빌드
if [ $( docker ps -a | grep sgbg-server | wc -l ) -gt 0 ]; then
  docker stop sgbg-server
  docker rm sgbg-server
fi
if [ $( docker images | grep sgbg-server | wc -l ) -gt 0 ]; then
  docker image rm sgbg-server
fi

docker ps
docker ps -a

cd ..
docker cp jenkins:/var/jenkins_home/workspace/application.properties SGBG/backend/src/main/resources
docker cp jenkins:/var/jenkins_home/workspace/keystore.p12 SGBG/backend/src/main/resources

docker build -t sgbg-server:latest -f SGBG/backend/Dockerfile .
docker run --name sgbg-server -d -p 8080:8080 sgbg-server
```

- `application.properties` 와 `keystore.p12` 같은 secret file들은 gitlab에 올리지 않아서, docker cp 를 통해 미리 복사 작업을 진행

Build Step 2: Frontend 빌드

```
docker cp jenkins:/var/jenkins_home/workspace/.env SGBG/frontend

# 프론트 디렉토리 진입
cd frontend

# npm build
yarn install
yarn build

# 결과물 nginx 컨테이너에 복사
docker cp build sgbg-nginx:/usr/share/nginx/html
```