Porting Menual

개발환경

Deploy Server

• Ubuntu 20:04:LTS

Front-end

Vue

vue : 2.6.11vue/cli : 4.5.0

IDE

• IntelliJ: 2019.3.5

Visual Studio Code : 1.59.0MySqlWorkbench : 8.0.21

DB

mysql: 8.0.25redis-server: 5.0.7

Back-end

• java : openjdk version "1.8.0_262"

• Spring

• spring boot 2.5.2

• gradle-6.8.3

• npm: 6.14.13

docker

• docker: 20.10.7

• docker-compose version 1.28.0-rc

배포방법

먼저 배포할 서버에 접속해 있다고 가정하여 진행하겠습니다. 설명할 서버 환경은 Ubuntu 20.04:LTS 기준입니다.

Java 8 버전 설치

1. Azul의 public key 추가

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 0x219BD9C9
```

2. Azul Repository 추가

```
{\tt sudo \ apt-add-repository \ 'deb \ http://repos.azulsystems.com/ubuntu \ stable \ main'}
```

3. apt-get 업데이트

```
apt-get update
```

4. zulu-8 설치

```
apt-get install zulu-8
```

5. 환경변수 설정

```
# /etc/profile
export JAVA_HOME=/usr/lib/jvm/zulu-8-amd64
```

6. 자바 설치확인

```
java -version
```

Docker-CE설치

1. apt 라이브러리 update 하기

```
sudo apt update
```

2. 다운로드를 위한 util 설치

```
sudo apt-get install \
   apt-transport-https \
   ca-certificates \
   curl \
   gnupg-agent \
   software-properties-common
```

3. Docker 공식 GPG 키 추가하기

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. apt 리스트에 도커 다운경로 추가하기

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

5. 도커 다운로드

```
apt-cache policy docker-ce
```

6. 도커 설치하기

```
sudo apt install docker-ce
# apt 패키지 인텍스 업데이트하기
sudo apt-get update
```

7. sudo 없이 도커를 사용하기 위해 사용자를 docker 그룹에 등록

```
sudo usermod -aG docker [현재 사용자]
# 이를 반영하기 위한 시스템 재부팅
sudo reboot
```

8. docker-ce를 설치했다면 이제 docker-compose도 설치해주자

```
sudo\ curl\ -L\ https://github.com/docker/compose/releases/download/1.28.0-rc2/docker-compose-`uname\ -s`-`uname\ -m`\ -o\ /usr/local/bin/docker/compose-`uname\ -m`\ -o\ /usr/local/bin/docker/compose-`uname\ -s`-`uname\ -s`-`uname\ -m`\ -o\ /usr/local/bin/docker/compose-`uname\ -s`-`uname\ -s`-`uname\ -m`\ -o\ /usr/local/bin/docker/compose-`uname\ -s`-`uname\ -s`-`un
```

9. docker-compose도 실행권한을 주자 (다른 방식 연습!)

```
sudo chmod +x /usr/local/bin/docker-compose
```

Mysql 설치

1. Mysql 설치

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

- 2. ubunut mysql은 대소문자 구별이 default로 설정되어 있음.
 - 이를 해결하려면 아래의 과정을 수행한다.
 - 1. Mysql 서비스 중지

```
sudo service mysql stop
```

2. Mysql 데이터 디렉토리 삭제

```
sudo rm -rf /var/lib/mysql
```

3. Mysql 데이터 디렉토리 재생성

```
sudo mkdir /var/lib/mysql
sudo chown mysql:mysql /var/lib/mysql
sudo chmod 700 /var/lib/mysql
```

4. lower_case_table_names = 1를 [mysqld] 섹션에 추가

```
# /etc/mysql/mysql.conf.d/mysqld.conf
[mysqld]
...
lower_case_table_names = 1
```

5. Mysql 초기화

```
sudo mysqld --defaults-file=/etc/mysql/my.cnf --initialize --lower_case_table_names=1 --user=mysql --console
```

6. Mysql 서비스 시작

```
sudo service mysql start
```

7. 새로 생성된 root의 비밀번호 검색

```
sudo grep 'temporary password' /var/log/mysql/error.log
```

8. root로 접속

```
sudo mysql -u root -p
```

9. 비밀번호 변경

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '1234';
```

10. lower_case_table_names을 해서 1 출력 확인

```
SHOW VARIABLES LIKE 'lower_case_%';
```

- 3. AWS 설정에서 보안그룹 설정을 통해 mysql 기본 포트(3306)을 열어주어야 한다.
- 4. 외부접속 허용하기 ⇒ mysqld.cnf 수정

```
# /etc/mysql/mysql.conf.d/mysqld.cnf
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

# bind-address를 찾아서 0.0.0.0으로 수정
bind-address = 0.0.0.0
```

5. mysql 재시작 및 접속

```
sudo mysql -u root -p
```

6. 외부접속 권한을 가진 유저생성

```
CREATE USER 'root'@'%' IDENTIFIED BY '1234';

GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

• MYSQL 8.0부터 Grant 명령어를 사용하여 묵시적으로 사용자를 생성할 수 없다. 따라서 '%' : 원격접속을 할 수 있는 사용자(여기서는 root)를 생성해 주어야한다.

소스코드 Git clone 받기

git clone https://lab.ssafy.com/s05-webmobile2-sub3/S05P13A602.git

SSL 인증서 발급

SSL For Free

1. SSL 발급받을 도메인 입력하기

https://www.sslforfree.com/ ← 접속



몇 분 만에 무료 SSL 인증서 및 와일드카드 SSL 인증서

음 보안 | https://

naver.com

무료 SSL 인증서 생성



영원히 100% 무료

다시는 SSL 비용을 지불하지 마십시오. 무료 90일 인증서와 함께 ZeroSSL 로 구동됩니다 .



널리 신뢰받는

당사의 무료 SSL 인증서는 전 세계 모든 주요 브라우저의 99.9%에서 신뢰됩니다.



SSL 혜택 즐기기

사용자 정보를 보호하고 신뢰를 구축하며 검색 엔진 순위를 향상시킵니다.

2. 회원가입

가입하기

SSL 인증서를 만들고 관리하려면 무료 계정에 가입하세요.

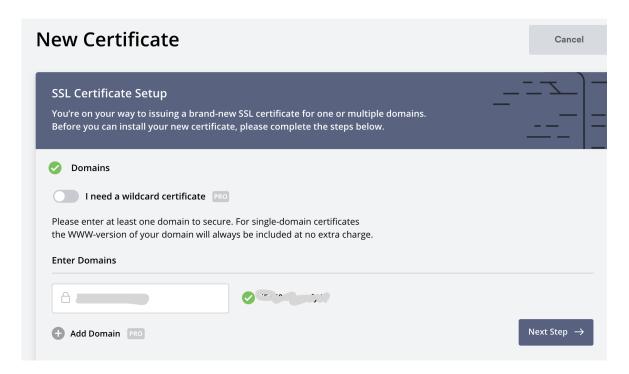
도메인:
이메일 주소
xxxxxxx@gmail.com
비밀번호
•••••
등록하다

중요한 서비스 업데이트를 수신하는 데 동의합니다.

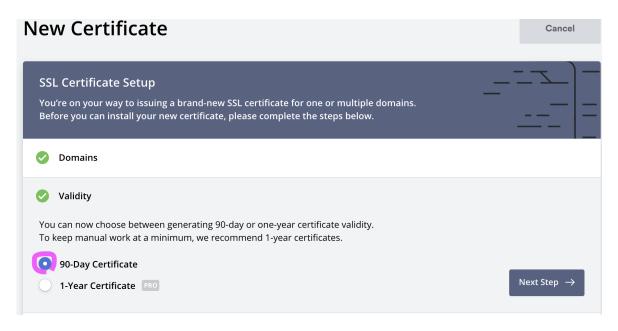
로그인

3. 인증서 신청

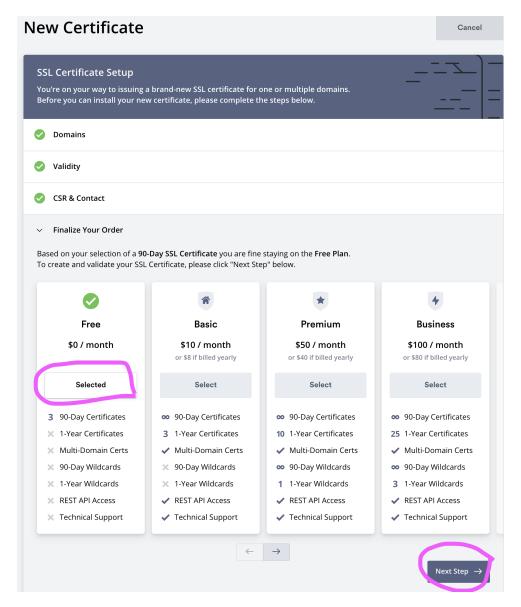
a. 도메인 입력



b. 인증 날짜 90일 설정



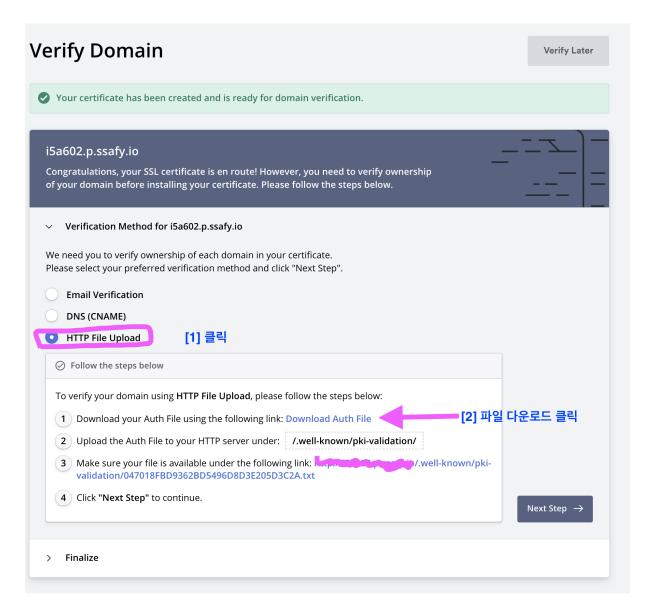
c. Free 선택 후 Next Step



4. 도메인 소유권 인증

이제 해당 도메인이 내가 소유하고 있다는 것을 증명해야한다. 그 이유는 우리가 막 naver같은 도메인에 SSL의 인증서를 발급받는 것을 막기 위함

a. 인증 txt 다운로드



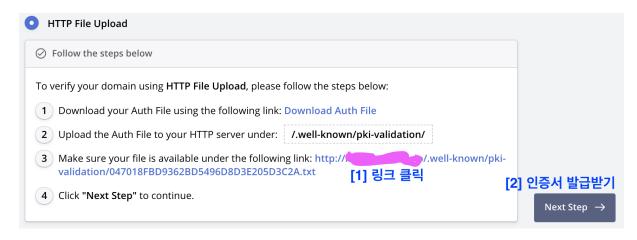
b. .txt 파일 복사하기

S05P13A602/.nginx/zerossl 폴더 밑에 .txt파일을 복사

c. nginx 실행

docker-compose up

d. 인증확인



- 링크를 클릭하여 정상적으로 .txt 파일의 내용이 보이는지 확인을 합니다.
 - 그리고 인증서 발급창으로 이동합니다.
- 5. 인증서 발급받기
- Verify Domain 버튼을 눌러 도메인을 인증
- 성공한다면, Server Type을 Nginx로 선택한후, 인증서를 다운로드 받습니다.
- 6. 인증서 복사하기

S05P13A602/.nginx/cert 폴더에 저장

Script 실행

cd S05P13A602 bash start.sh

- Docker를 사용하여 build및 실행시키는 단계입니다.
- start.sh 쉘 스크립트안에 명령어가 내장되어 있으며, 총 3개의 컨테이너를 데몬으로 실행시키게 됩니다.
 - nginx : reverse proxy를 담당하는 nginx
 - client : Vue Application
 - · server : Spring Boot Application

Docker

```
# start.sh

# Spring gradle build
cd commbServer && ./gradlew bootJar && cd ..

# 도커이미지 최신화
docker-compose pull

# 도커컴포즈 빌드 및 데몬 실행
COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker-compose up --build -d

# 더미이미지 삭제
docker rmi $(docker images -f "dangling=true" -q) -f
```

```
#docker-compose.yml
version: '3'
services:
 # 스프링 컨테이너
  server:
    container_name: server
    build:
     context: ./commbServer
       DOMAIN: i5a602.p.ssafy.io
      - "8080:8080"
  # vue 컨테이너
  client:
    container_name: client
    build: ./client
   ports:
      - "8000:8000"
    depends on:
      - server
  # nginx 컨테이너
  nginx:
    container_name: nginx
    # image: proxy
    build: ./.nginx
    depends_on:
     - server
- client
    volumes:
     - .nginx/conf.d:/etc/nginx/conf.d # 설정정보
- .nginx/zerossl:/var/www/zerossl/.well-known/pki-validation # SSL 도메인 인증
      - .nginx/cert:/cert # SSL 인증서 보관장소
    ports:
      - 80:80 # for Http
      - 443:443 # for Https
```

• nginx 설정파일

경로: S05P13A602/.nginx/conf.d/default.conf

```
server {
              80; # 80포트 허용
   listen
   server_name i5a602.p.ssafy.io; # 서버이름
   # reverse proxy
   location /api { # /api 요청시 spring으로
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_pass http://server:8080;
  # 이경로로 요청이오면 SSL 도메인 확인 txt 반환
   location /.well-known/pki-validation/ {
      allow all;
       root /var/www/zerossl;
   location / { # 나머지는 vue로
       proxy_pass http://client:8000;
server {
   listen 443 ssl; # 443 포트허용
   server_name i5a602.p.ssafy.io;
   proxy_connect_timeout 1d;
   proxy_send_timeout 1d;
   proxy_read_timeout 1d;
```

```
# 위와동일
location /api {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://server:8080;
}

location / {
    proxy_pass http://client:8000;
}

# SSL 인종설정
ssl_certificate /cert/certificate.crt;
ssl_certificate_key /cert/private.key;
}
```

배포시 주의사항

- 1. 현재 모든 경로가 하드코딩되어 있으므로 도메인을 변경 혹은 localhost로 실행할 경우, 하드코딩된 경로수정필요
 - 수정할 문자열
 - <u>i5a602.p.ssafy.io</u> → localhost
 - https → http
 - applicaion.yaml 수정

```
# 디비수정
spring:
datasource:
    url : jdbc:mysql://i5a602.p.ssafy.io/Commb?serverTimezone=Asia/Seoul
    username : root
    password : 1234

# redis 수정
redis:
    host : i5a602.p.ssafy.io
    password : 1234
    port : 6379

# dynamic 경로수정
dynamic:
    path: https://i5a602.p.ssafy.io/
front:
    path: https://i5a602.p.ssafy.io/
```

• .env.production 수정

```
VUE_APP_SERVER_URL=https://i5a602.p.ssafy.io/api
```

- 2. Https 주의
 - FCM(Firebase Cloud Messaging) 서비스를 사용중인데,
 localhost 이외의 환경에서 https가 아닐경우 제대로 실행되지 않는다!
 - 따라서, 특정 도메인에 배포할경우 꼭 SSL 인증서가 필요함.

데이터베이스 접속정보

• 경로: S05P13A602/commbServer/src/main/resources/application.yaml

Mysql

• host : <u>i5a602.p.ssafy.io</u>

port : 3306user : rootpassword : 1234

Redis

• host : i5a602.p.ssafy.io

port: 6379password: 1234

외부서비스 필요정보

• 경로 : S05P13A602/commbServer/src/main/resources/application.yaml

Kakao API

```
spring:
 security:
      oauth2:
        client:
          registration:
             kakao:
              authorization\hbox{-}grant\hbox{-}type\hbox{: }authorization\hbox{\_}code
              client-id: xxxxxxx
              client-secret: xxxxxx
              redirect-uri: "{baseUrl}/api/login/oauth2/code/{registrationId}"
              scope:
              client-authentication-method: POST
              client-name: Kakao
          provider:
             kakao:
              authorization-uri: https://kauth.kakao.com/oauth/authorize
              token-uri: https://kauth.kakao.com/oauth/token
              user-info-uri: https://kapi.kakao.com/v2/user/me
              user-name-attribute: id
```

SMTP

```
spring
mail:
    host: smtp.gmail.com
port: 587
    username: xxxx
password: xxxxx
properties:
    mail:
    smtp:
    auth: true
    starttls:
    enable: true
    required: true
    connectiontimeout: 5000
    timeout: 5000
    writetimeout: 5000
```

S3 Buket

```
cloud:
    aws:
    credentials:
    accessKey: xxxxx
    secretKey: xxxxxx
s3:
    bucket: ${buket_id}
    region:
    static: ap-northeast-2
stack:
    auto: false
```

Firebase Cloud Messaging

• config.js