포팅 매뉴얼

개발 환경

Deploy Server

- Ubuntu 20:04:LTS

Frontend

- Vue : 2.6.11
- vue/cli : 4.5.13

DB

- MySQL : 8.0.27
- redis : 5.0.7

Backend

- java : openjdk version "1.8.0_262"
- Spring
  - Spring Boot : 2.5.5
  - gradle : 6.8.3
- npm : 6.14.13
- docker : 20.10.8
- docker-compose : 1.28.0-rc2

IDE

- Visual Studio Code : 1.61.2
- IntelliJ : 2019.3.5
- STS : 3.9.14.RELEASE
- MySQLWorkBench : 8.0.25
- Unity : 2020.3.20f1

배포 방법

> 배포할 서버에 접속해 있다고 가정

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

```
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

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

6. 자바 설치 확인

```
java -version
```

Docker-CE 설치

1. apt 라이브러리 업데이트

```
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 인덱스 업데이트하기
sudo apt-get update
```

7. sudo 없이 도커를 사용하기 위해 사용자를 docker 그룹에 등록

```
sudo usermod -aG docker [현재 사용자]

# 반영하기 위한 시스템 재부팅
sudo reboot
```

8. docker-compose도 설치

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

9. 실행권한 주기

```
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 구별 해결 과정
   a. Mysql 서비스 중지

```
sudo service mysql stop
```

   b. Mysql 데이터 디렉토리 삭제

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

   c. Mysql 데이터 디렉토리 재생성

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

   d. lower_case_table_names = 1를 [mysqld] 섹션에 추가

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

   e. Mysql 초기화

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

   f. Mysql 서비스 시작

```
sudo service mysql start
```

g. 새로 생성된 root의 비밀번호 검색

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

h. root로 접속

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

i. 비밀번호 변경

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

j. lower_case_table_names을 해서 1 출력 확인

3. 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 service mysql restart
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.

Redis 설치

1. redis-server설치

```
sudo apt install redis-server
```

2. redis.conf 편집

```
sudo vi /etc/redis/redis.conf


/requirepass
 -> requirepass [password]

/bind
 -> bind 0.0.0.0 ::1

maxmemory 500m
```

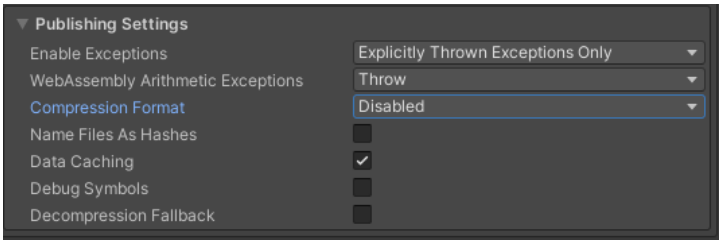
```
maxmemory-policy allkeys-lru

:wq!
```

3. 업그레이드 → 종료 → 재시작

```
sudo apt update && sudo apt upgrade
sudo service redis-server stop
sudo service redis-server start
```

Unity Webgl Build

1. unity에서 프로젝트의 unity 폴더를 import
2. File → Build setting 에서 Webgl 로 Switch Platform
3. Player settings > Player > Public Settings 에서 다음과 같이 설정



4. Build를 누르고 frontend/public 에서 untiy 폴더를 만들고 build한다.

소스코드 Git clone 받기

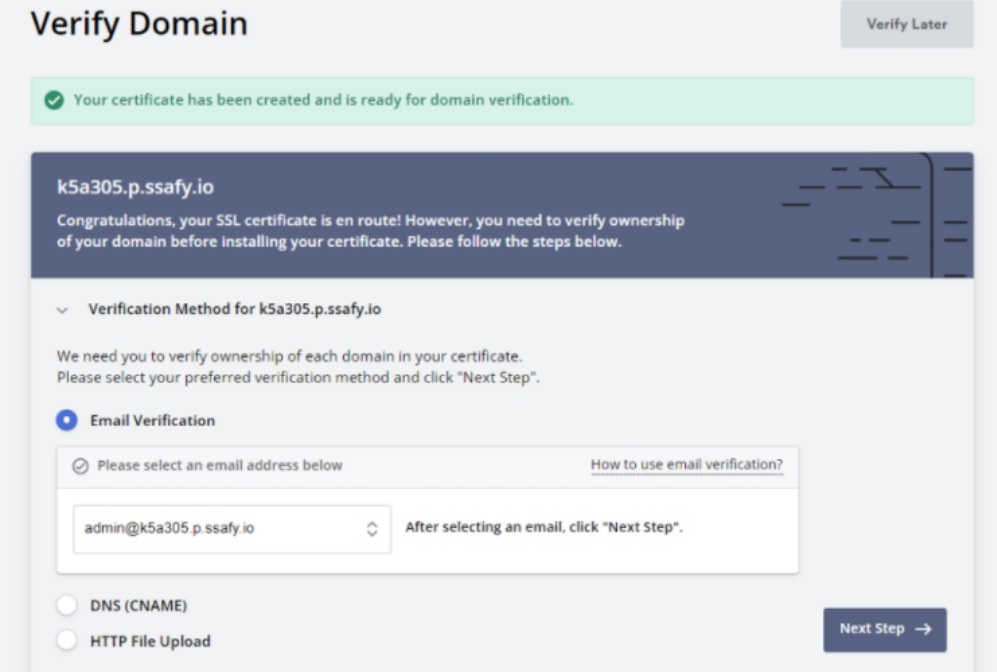
```
git clone https://lab.ssafy.com/s05-final/S05P31A305.git
```

SSL 인증서 발급

SSL For Free

- 90일간 무료 인증서를 받음

1. SSL 발급받을 도메인 입력하기
   https://www.sslforfree.com/ ← 접속
2. 회원가입 후 도메인 인증 → HTTP File Upload 인증 방식 선택

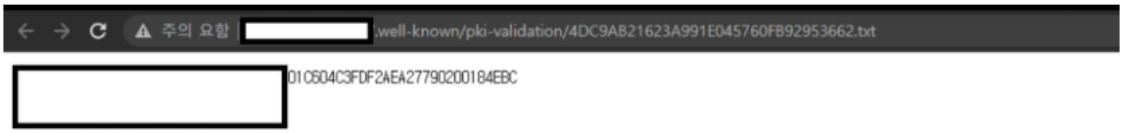


3. Download Auth File을 클릭하여 txt 파일을 다운 받고 frontend\.nginx\zerossl 폴더를 생성하여 저장

.4. ngnix 실행

```
docker-compose up
```

5. 제공된 링크를 눌러 txt 내용이 보이는지 확인



6. 인증서 발급받기
   Verify Domain → Server Type을 Nginx로 선택 후, 인증서를 다운로드 받습니다.
7. 인증서 복사하기
   project/frontend/.nginx/cert 폴더에 저장
8. 서버에서 git pull 받은 후 다음을 실행하여 .crt 파일 병합

```
sudo vi new_certificate.crt
:wq!

sudo chmod +777 new_certificate.crt
sudo cat certificate.crt ca_bundle.crt >> new_certificate.crt
sudo chmod 644 new_certificate.crt

# 병합 순서 중요함. 반대로 병합 시 private.key로 풀 수 없음
```

9. nginix 재시작

```
sudo /etc/init.d/nginx restart
sudo netstat -nltp | grep 80
sudo kill -9 [PID]
```

Gradle Build

1. Gradle 설치

```
sudo apt install gradle
```

2. 프로젝트에서 gradlew 사용하여 빌드

```
cd Backend

chmod +x gradlew
sudo ./gradlew build
```

Docker Image 빌드 및 실행

아래의 docker file 관련 모두 작성 후 실행

```
cd S05P31A305$
docker -compose pull
docker -compose up --build -d
```

Docker file 정보

- docker-compose.yml

```
version: '3'

services:
  server:
    container_name: back
    build:
      context: ./Backend
    ports:
      - "8080:8080"

  client:
    container_name: front
    build:
      context: ./frontend
    depends_on:
      - server
    ports:
      - "8000:8000"

  nginx:
    container_name: nginx
    build: ./frontend/.nginx
    depends_on:
      - server
      - client
    volumes:
      - ./frontend/.nginx/conf.d:/etc/nginx/conf.d
      - ./frontend/.nginx/zerossl:/var/www/zerossl/.well-known/pki-validation
      - ./frontend/.nginx/cert:/cert
    ports:
      - 80:80
      - 443:443
```

- Backend/Dockerfile

```
FROM openjdk:8-jdk-alpine as builder

WORKDIR application

ARG JAR_FILE=build/libs/ssafy-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} application.jar

EXPOSE 8080

RUN java -Djarmode=layertools -jar application.jar extract

FROM openjdk:8-jdk-alpine
WORKDIR application
ARG DOMAIN
ENV port 8080
ENV spring.profiles.active local
COPY --from=builder application/dependencies/ ./
COPY --from=builder application/spring-boot-loader/ ./
COPY --from=builder application/snapshot-dependencies/ ./
COPY --from=builder application/application/ ./
RUN echo "$DOMAIN"
ENV DOMAIN=$DOMAIN
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher"]



#RUN java -Djarmode=layertools -jar application.jar extract
#
FROM openjdk:8-jdk-alpine
WORKDIR application
ARG DOMAIN
ENV port 8080
ENV spring.profiles.active local
COPY --from=builder application/dependencies/ ./
COPY --from=builder application/spring-boot-loader/ ./
COPY --from=builder application/snapshot-dependencies/ ./
COPY --from=builder application/application/ ./
RUN echo "$DOMAIN"
ENV DOMAIN=$DOMAIN
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher"]
# ENTRYPOINT ["java", "-jar", "/application.jar"]
```

- frontend/Dockerfile

```
FROM node:lts-alpine as build-stage

WORKDIR /app

COPY package*.json ./

RUN npm install
RUN npm install vue-router

COPY . .

RUN npm run build

FROM nginx:stable-alpine as production-stage
COPY ./.nginx/nginx.conf /etc/nginx/conf.d/default.conf
RUN rm -rf /usr/share/nginx/html/*

COPY --from=build-stage /app/dist /usr/share/nginx/html

EXPOSE 8000

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

- frontend/.nginx/ngnix.conf

```
server{
   listen 8000;
   client_max_body_size 5M;
```

```
    location / {
      alias /usr/share/nginx/html/;
      try_files $uri $uri/ /index.html;
    }
}

types {
    application/wasm wasm;
}
```

- frontend/.nginx/conf.d/default.config

```
server {
  listen 80;

  location /api {
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_pass http://k5a305.p.ssafy.io:8080;
  }

  location /.well-known/pki-validation/ {
    allow all;
    root /var/www/zerossl;
  }

  location / {
    proxy_pass http://k5a305.p.ssafy.io:8000;
  }
}

server{
  listen 443 ssl;

  proxy_connect_timeout 1d;
  proxy_send_timeout 1d;
  proxy_read_timeout 1d;

  ssl_certificate /cert/new_certificate.crt;
  ssl_certificate_key /cert/private.key;

  location /api {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://k5a305.p.ssafy.io:8080;
  }

  location / {
    proxy_pass http://k5a305.p.ssafy.io:8000;
  }
}
```

데이터베이스 정보

MySQL

- HOST : 13.125.29.233(k5a305.p.ssafy.io)
- port : 3306
- user : root
- password : k5a_meta

Redis

- HOST : k5a305.p.ssafy.io
- port : 6379

접속 URL

- https://k5a305.p.ssafy.io

외부 서비스 필요 정보

Kakao API(카카오 로그인)

헤더 클라이언트 id 정보

```
const kakaoHeader = {
  Authorization: "xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "Content-type": "application/x-www-form-urlencoded;charset=utf-8",
};

const data = {
  grant_type: "authorization_code",
  client_id: "xxxxxxxxxxxxxxxxxxx",
  redirect_uri: "https://k5a305.p.ssafy.io" + "/auth",
  code: code,
};
```

SMTP(이메일 인증)

```
spring:
  mail:
      host: smtp.gmail.com
      port: 587
      username: xxxxxxxxxx
      password: xxxxxxxxxx
      properties:
        mail.smtp.auth: true
        mail.smtp.starttls.enable: true
```

Firebase(FCM and Chatting)

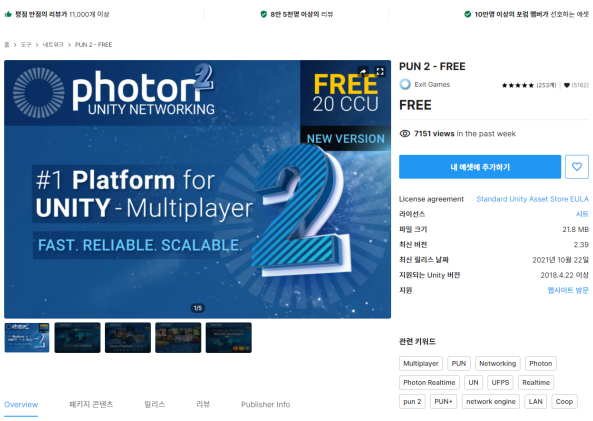
- config

```
{
  apiKey: "xxxxxxxxxxxxxxxxxxxxxxxxx",
  authDomain: "xxxxxxxxxxx.firebaseapp.com",
  databaseURL: "https://xxxxxxxxxxxx.firebaseio.com",
  projectId: "xxxxxxxxxxxxxx",
  storageBucket: "favorable-bolt-113915.appspot.com",
  messagingSenderId: "xxxxxxxxxxx",
  appId: "1:xxxxxxxxxxxxxxxxxxxxxxxxx:web:xxxxxxxxxxxxxxxxxxxxx",
  measurementId: "xxxxxxxxxxxxxxxxxxxxxx",
};
```

- Cloud messaging 웹 푸시 인증서
- firebase Sdk path

Photon PUN2 (Unity 게임서버 멀티플레이 환경 구축)

1. Unity Asset Store에서 PUN2 다운로드



2. Photon server instance 생성

3. Unity 프로젝트에 PUN2 import 후 setup에 AppId 입력

✨ 주의할 점

→ Photon server의 무료 버전은 최대 동시 접속자 수 제한있음!

주의 사항

- FCM(Firebase Cloud Messaging) 이용시 localhost 이외의 환경에서 https가 아닐경우 실행 되지 않으므로 특정 도메인에 배포할 경우 꼭 SSL 인증서가 필요함.
- 서버에서 git pull, push 시 **error: cannot open .git/FETCH_HEAD: Permission denied** 발생한 경우 다음을 실행

```
sudo chown -R $USER:
```