포팅메뉴얼

개발환경

1.1 개발 환경

- JAVA 11
- SpringBoot : 2.7.13
- Gradle : 8.1.1
- Node.js : 14.17.0
- mysql : 5.7.41

1.2. 환경 변수

- **프론트엔드 환경변수 (.env)**

```
NEXT_PUBLIC_SERVER_URL=https://k9c109.p.ssafy.io/api/v1/
NEXT_PUBLIC_GOOGLE_LOGIN_PATH=https://k9c109.p.ssafy.io/oauth2/authorization/google
NEXT_PUBLIC_GOOGLE_MAPS={google map key}
NEXT_PUBLIC_WS_BASE_URL=https://k9c109.p.ssafy.io/ws
NEXT_PUBLIC_DEVELOP_WS_BASE_URL=http://localhost:8080/ws
NEXT_PUBLIC_YOUTUBE_URL=https://www.googleapis.com/youtube/v3
NEXT_PUBLIC_YOUTUBE_KEY={youtube data api key}
```

- **백엔드 환경 변수 (.env)**

```
MYSQL_ROOT_PASSWORD={mysql pw}
MYSQL_USER={mysql id}
MYSQL_PASSWORD={mysql pw}
S3_ACCESS_KEY={s3 key}
S3_SECRET_KEY={s3 key}
CLIENT_ID={google console client id}
CLIENT_SECRET={google console client key}
REDIS_PASSWORD={redis pw}
REDIS_HOST=k9c109.p.ssafy.io;
JWT_SECRET={jwt key}
GEOCODER_API_KEY={geocoder key}
WEATHER_SERVICE_KEY={data.go.kr key}
OPENAPI_KEY={openai key}
YOUTUBE_API_KEY={youtube data api key}
```

1.3 외부 서비스

- Google Cloud Vision
- OpenAI API
- Youtube Data Api
- Youtube Iframe Api

1. EC2 접속

host `k9c109.p.ssafy.io`

user `ubuntu`

port `22`

2. apt, apt-get 최신화

`sudo apt update`

`sudo apt upgrade`

`sudo apt-get update`

`sudo apt-get upgrade`

3. nginx, ssl, 방화벽 세팅

nginx

- `sudo apt install nginx`
- 주요 명령어
  - 상태확인 `sudo systemctl status nginx`
  - 서비스 시작 `sudo systemctl start nginx`
  - 서비스 중지 `sudo system stop nginx`
  - 환경 설정 `sudo vi /etc/nginx/sites-available/{파일명}.conf`

SSL

- let’s Encrypt 설치
  - `sudo apt-get install letsencrypt`
- certBot 설치
  - `sudo apt-get install certbot python3-certbot-nginx`
  - 활성화 `sudo certbot --nginx`
  - 도메인 추가 sudo certbot --nginx -d {domain}

방화벽

상태확인 `sudo ufw status`

방화벽 사용 설정 `sudo ufw enable`

방화벽 사용 해제 `sudo ufw disable`

개방했던 포트 삭제 `sudo ufw delete {포트번호}`

22번 포트 개방 `sudo ufw allow 22`

80번 포트 개방 `sudo ufw allow 80`

ssh 기본 포트(22) 개방 `sudo ufw allow ssh`

http 기본 포트(80) 개방 `sudo ufw allow http`

https 기본 포트(443) 개방 `sudo ufw allow https`

4. docker 설치

```
# HTTPS를 통해 리포지토리를 사용할 수 있도록 패키지 인덱스를 업데이트 (apt)하고
# 패키지를 설치
```

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

# Docker의 공식 GPG 키를 추가
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# 리포지토리를 설정
echo \
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 패키지 index를 업데이트
sudo apt-get update

# Docker Engine, containerd 및 Docker Compose를 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 이미지를 실행하여 Docker 엔진 설치가 성공했는지 확인
sudo docker run hello-world
docker -v
```

docker 명령어

`docker run --name {name} -d -p 3306:3306 {image-name}`

- {image-name}으로 {name} 컨테이너를 만들거야,
- 백그라운드(-d)로 포트포워딩(-p)은 3306에서 3306으로.
- 추가옵션
  - -v : 볼륨마운트
  - -e : 환경변수 만들기

docker-compose

`sudo apt install docker-compose`

docker 프론트엔드 배포

next 프로젝트 루트경로에 Dockerfile 작성.

```
# 의존성 설치
FROM node:alpine AS deps
RUN apk add --no-cache libc6-compat python3 build-base
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci

# next 프로젝트 빌드
FROM node:alpine AS builder
WORKDIR /app
COPY . .
COPY --from=deps /app/node_modules ./node_modules
RUN npm run build
RUN npm ci --production
RUN rm -rf ./.next/cache

# 파일 copy 후, next 프로젝트 실행
FROM node:alpine AS runner
WORKDIR /app

ENV NODE_ENV=production
ENV SHOULD_PROFILE=true
ENV SHOULD_TRACE=true

RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001

COPY --from=builder /app/package.json ./package.json
```

```
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder --chown=nextjs:nodejs /app/.next ./.next
COPY --from=builder /app/public ./public
COPY --from=builder /app/next.config.js ./next.config.js

USER nextjs

# NginX config
RUN if [ -f /etc/nginx/conf.d/default.conf ]; then \
        rm /etc/nginx/conf.d/default.conf; \
    fi
COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 3000

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

docker image build & push

`docker build -t wjs5025/play-place:0.1.0 .`
`docker push wjs5025/play-place:0.1.0`

5. mysql 설정

EC2 서버에 그대로 설치할 경우

1. MySQL 설치 ⇒ `sudo apt install mysql-server`
2. MySQL 상태 확인 ⇒ `sudo systemctl status mysql`
3. root 계정 접속 ⇒ `sudo mysql -u root -p`
4. mysql로 DB 변경 ⇒ `use mysql;`
5. 계정 생성 ⇒ `create user ‘아이디’@’호스트’ identified with mysql_native_password by ‘비밀번호';`
6. 변경 사항 적용 ⇒ `flush privileges;`
7. 계정 생성 확인 ⇒ `select user, host from user;`
8. DB 생성 및 확인 ⇒ `create database {DB명}; show databases;`
9. 권한 부여(변경 사항 적용 필요) ⇒ `grant all privileges on {DB스키마}.{권한} to ‘아이디’@’호스트’;`
10. 외부 접속 허용하기 ⇒ `sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf`
    - bind-addredd 값을 특정 IP로 수정, ssh 터널링을 이용한 접속, 0.0.0.0 으로 설정할 경우, 해킹 위험

root 비밀번호 변경하기

```
mysql 접속 후,
alter user 'root'@'localhost' identified with mysql_native_password by '변경할 비밀번호';
```

docker 컨테이너로 띄우기

`docker pull mysql`

- 특정 버전으로 띄우고 싶으면, `docker pull mysql:8.0.22`
- 버전 정보는 Docker Hub에서… https://hub.docker.com/

`docker images`

- 도커 이미지 목록 확인
- mysql 이 잘 받아졌나 확인.

`sudo service mysql stop`

- 도커 컨테이너 만들기 전에, 돌고 있는 mysql 서비스 종료하기. (포트 중복 때문에)

`docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mysql:latest`

- mysql 도커 돌리자

`docker ps`

- 잘 돌고 있나 확인해

`docker exec -it mysql-container bash`

- mysql docker 컨테이너에 접속

`mysql -u root -p`

- mysql 에 접속

docker 명령어.

현재 돌고 있는 컨테이너 목록 `docker ps`

실행했던 컨테이너 목록 `docker ps -a`

중지 `docker stop {container-name}`

시작 `docker start {container-name}`

재시작 `docker restart {container-name}`

Redis

레디스 컨테이너 띄우기

`sudo docker run --rm -d -p 6379:6379 --name redis redis`

레디스 쉘 접속

`docker exec -it redis redis-cli`

비번변경

`config set requirepass wjsdlsgur1!`

6. 젠킨스 설정 (CI/CD)

1. jenkins 이미지 내려받기

이미지 받기 `sudo docker pull jenkins/jenkins:lts`

이미지 확인 `sudo docker images`

이미지 띄우기

```
sudo docker run -d --privileged=true --env JENKINS_OPTS=--httpPort=8081 -p 8081:8081 -v jenkins:/var/jenkins_home -v
/usr/bin/docker:/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock --name jenkins -u root jenkins/jenkins:lts
```

- 임시

```
sudo docker run -d -p 8080:8080 -v /jenkins:/var/jenkins_home --name jenkins -u root jenkins/jenkins:lts
```

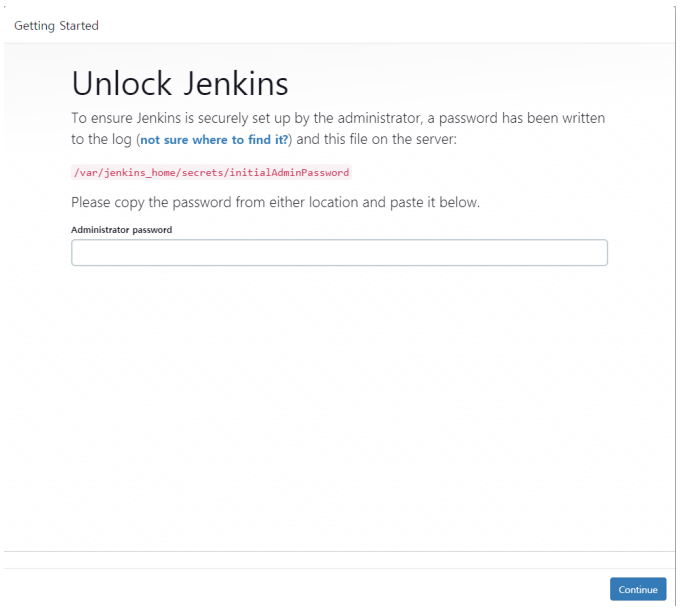
- 명령어 해독타임!
  - `-d` : 컨테이너를 데몬(백그라운드)으로 띄운다.
  - `-p 8080:8080` : 포트포워딩. (좌측이 호스트 포트, 우측이 컨테이너 포트)
  - **-`v /jenkins:/var/jenkins_home` : 볼륨 마운트**
    - **도커 컨테이너의 데이터는 종료시 휘발되는데, 이를 보존하기 위한 방법.**
    - /var/jenkins_home 이라는 디렉토리를 호스트의 /jenkins와 마운트하고 데이터를 보존할 수 있음.
  - `--name jenkins` : 도커 컨테이너 이름 설정
  - `-u root` : 컨테이너가 실행될 리눅스 사용자 계정을 root로 명시.

2. jenkins로 접속

1. **브라우저에서, 서버의 8080 포트로 접속.**

   http://k9c109.p.ssafy.io:8080/

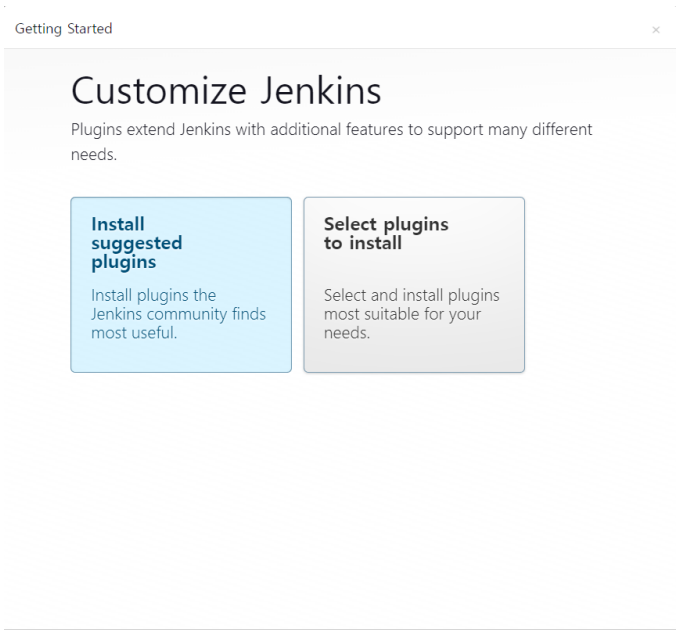
2. **인증번호 입력하기**

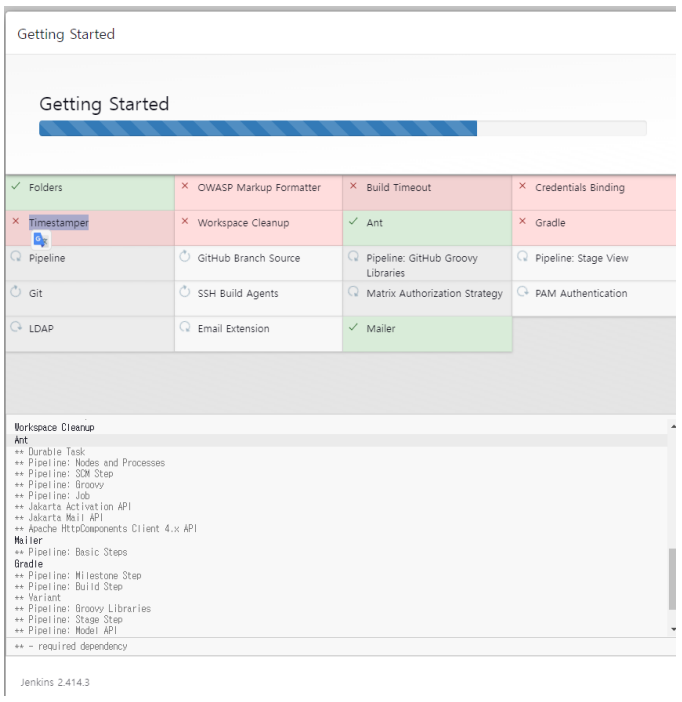


jenkins 컨테이너의 로그 확인 `sudo docker logs jenkins`

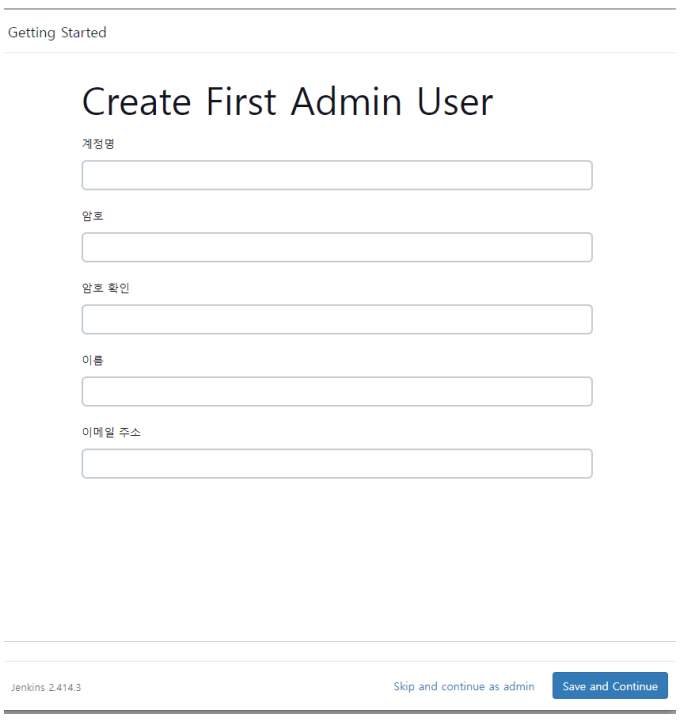
3. 플러그인 설치

   아래 화면에서 왼쪽꺼 클릭 (install suggested plugins)





4. 기본설정하기
    - 요구되는 정보들 입력



- 타임존 서울로 설정.
    - 젠킨스 접속 후, 설정 → Time Zone

Gitlab ↔ 젠킨스 연동

1. 깃랩 토큰 발급 받기.

- Settings → Access Tokens
- NewToken
- 정보 입력
    - Token name
    - Expiration date
    - Select a role (maintainer)
    - Select Scopes(all check!)
- 토큰 꼭 저장해두기. (페이지 이탈 시 다시 볼 수 업성…)

```
내 깃랩 젠킨스 토큰
UjQJsod1UsshSW1oyKnQ
```

2. 젠킨스에서 플러그인 설치

- 젠킨스 관리 탭 → 플러그인즈 → Available plugins 탭
- Deploy to container / Post build task / NodeJS(⇒ 프론트부분) / GitLab 설치

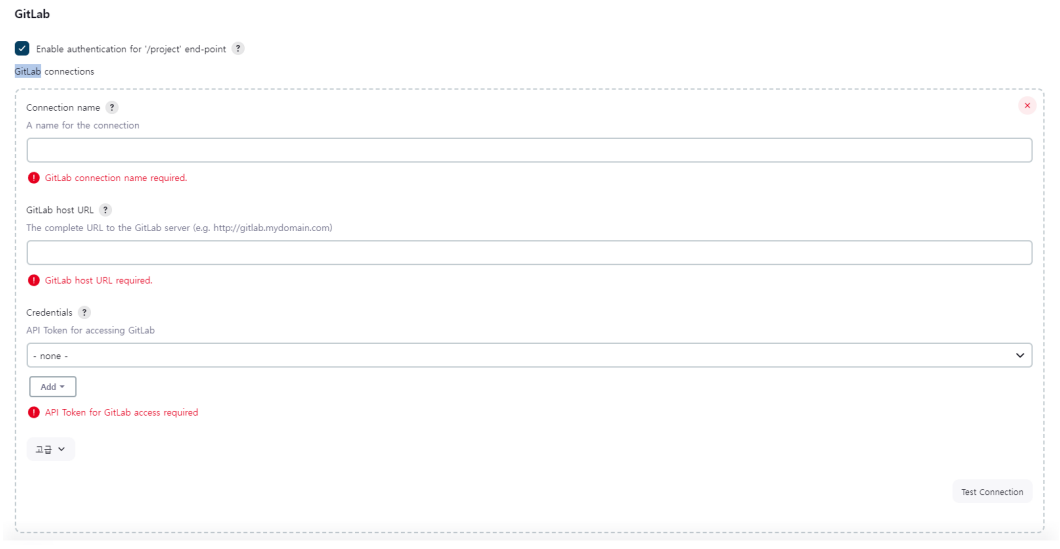
3. 인증 추가

- **젠킨스 관리 탭 → credentials → System → Global credentials (unrestricted) → Add Credentials**

- **credentials 추가 1**
  - Kind = Gitlab API token
  - Scope = global
  - API Token = 아까 복사해둔거.
  - id/description 은 적절하게.
- **credentials 추가 2**
  - kind = username with password
  - scope = global
  - username = 내 깃랩 아이디
  - password= 내 깃랩 패스워드

4. gitlab 연동

- jenkins 관리 → System ⇒ Gitlab 파트



- 정보 입력
  - connection name = 적당한거 입력
  - gitlab host url = https://lab.ssafy.com
  - credentials = gitlab api token(token name)

5. Item 생성

- 새로운 Item 생성 탭
- 이름 작성 후, Freestyle project 선택 후 OK
- 아이템의 콘솔에서, 소스코드 관리 탭에서 설정
  - 소스코드관리 = Git
  - Repository URL = Clone with HTTPS 레포 주소 입력.

    ```
    https://lab.ssafy.com/s09-final/S09P31C109.git
    ```

  - Branches to build = 기본 사용 브랜치명 사용

nginx 리버스 프록시 설정

etc/nginx/sites-avaliable/default

```
server  {
        listen [::]:80;
        server_name playplace.co.kr;

        location / {
                root /home/ubuntu/playplace;
                index index.html;
                try_files $uri /index.html;
        }

        # listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/playplace.co.kr/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/playplace.co.kr/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 {
        listen [::]:80;
        server_name k9c109.p.ssafy.io;

        location / {
                return 301 https://playplace.co.kr;
        }
```

```
        location /pp {
                proxy_pass http://localhost:3000;
                proxy_buffer_size               128k;
                proxy_buffers                   4 256k;
                proxy_busy_buffers_size         256k;

                proxy_set_header X-Real_IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
        }

        location /api/v1 {
                proxy_pass http://localhost:8080;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;

                # proxy_set_header Connection '';
                # proxy_http_version 1.1;
                # proxy_buffering off;
                # chunked_transfer_encoding off;
        }

        location /ws {
                proxy_pass http://localhost:8080;

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
        }


        location /oauth2 {
                proxy_pass http://localhost:8080$request_uri;
        }

        location /login/oauth2 {
                proxy_pass http://localhost:8080$request_uri;
        }

        # listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/k9c109.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/k9c109.p.ssafy.io/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
}
```