포팅매뉴얼

사용한 시스템 버전

버전

Aa 이름	☰ 태그
Unity	2021.3.11f1 LTS
node.js	16.17.0
jdk	11
spring boot	2.7.4
mysql	5.7

빌드 준비

1. Docker 다운로드 및 설정

사용할 서버에 도커를 설치해 준다. (ubuntu linux 사용)

```
# 기존의 도커를 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

sudo apt-get update

# https를 통해 리포지토리를 사용할수 있도록 패키지를 업데이트
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release

# /etc/apt/keyrings에 Docker의 공식 GPG키를 추가
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# 리포지토리 설정
echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker 설치

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io

# 설치 확인
docker -v

# 사용자 권한 설정
sudo usermod -aG docker $USER

# Docker-compose 설치
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2(다운받을버전)/docker-compose-$(uname -s)-$(uname -m)" -o /usr/lo
# 권한 설정
sudo chmod -x /usr/local/bin/docker-compose
# 설치 확인
docker-compose --version
```

2. Nginx Config 설정

```
server {
  listen 80;
  server_name k7a101.p.ssafy.io;

  location /.well-known/acme-challenge/ { # certbot
    root /var/www/certbot;
  }

  location / { # 아래 URL로 redirect
    return 301 https://k7a101.p.ssafy.io$request_uri;
  }
}

# 443번 포트로 들어오는 경우
server {
  listen 443 ssl;
  server_name k7a101.p.ssafy.io;

  ssl_certificate /etc/letsencrypt/live/k7a101.p.ssafy.io/fullchain.pem; #certbot
  ssl_certificate_key /etc/letsencrypt/live/k7a101.p.ssafy.io/privkey.pem; #certbot

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

  location / { # /로 들어오는 경우
    root /var/www/build;
              index index.html;
    try_files $uri $uri/ /index.html;
  }

  location /api { # /api로 들어오는 경우
    rewrite ^/api(/.*)$ $1 break; # /api -> / 로 rewrite
    proxy_pass https://k7a101.p.ssafy.io:8080; # 8080번 포트로 pass
    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 다운 및 설정

```
# jenkins volume 설정 파일 생성
sudo mkdir -p /home/ubuntu/jenkins

# jenkins 설치 및 실행 jenkins 볼륨과 docker 볼륨을 잡으며 8888포트로 실행한다.
sudo docker run \
--name jenkins \
-d \
-p 8888:8080 \
-p 50000:50000 \
-v /home/ubuntu/jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-u root \
jenkins/jenkins:lts


# jenkins container 내부에 docker를 설치
# jenkins container 접속
docker exec -it jenkins bash

# docker 설치
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
```

jenkins 설정하기

1. jenkins 관리의 플러그인 관리로 들어가 필요한 플러그인을 설치해준다.

gitlab, generic webhook trigger, gitlab api, gitlab authentication, docker, docker commons, docker pipeline, docker api, node 등

2. Global Tool Configuration에 들어가 사용하는 node버전을 맞춰 설정해준다.

3. 프로젝트 생성하기

생성할 프로젝트 이름을 적어 Freestyle project로 프로젝트를 생성해줍니다.

4. git 연결하기

소스코드 관리에서 연동할 git의 주소와 git에 연결할 수 있는 credential 유저를 적용시키고 branch Specifier에 CI/CD를 이용할 branch 를 적어준다

소스 코드 관리

○ None

● Git ?

Repositories ?

Repository URL ?   ✕

https://lab.ssafy.com/s07-final/S07P31A101.git

Credentials ?

nadadbsduf7@gmail.com/****** (final project CI / CD)   ⌄

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?   ✕

*/develop

5. 빌드 유발 및 Webhook 설정

webhook설정을 위해 고급을 눌러준다.

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☑ Build when a change is pushed to GitLab. GitLab webhook URL: http://k7a101.p.ssafy.io:8888/project/carum2 ?
  - Enabled GitLab triggers
    - ☑ Push Events
    - ☐ Push Events in case of branch delete
    - ☑ Opened Merge Request Events
    - ☐ Build only if new commits were pushed to Merge Request ?
    - ☐ Accepted Merge Request Events
    - ☐ Closed Merge Request Events

  - Rebuild open Merge Requests

Never	⌄

    - ☑ Approved Merge Requests (EE-only)
    - ☑ Comments

  - Comment (regex) for triggering a build ?

Jenkins please retry a build

  [고급...]

Comment (regex) for triggering a build ?

Jenkins please retry a build

- ☑ Enable [ci-skip]
- ☑ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

- ☑ Set build description to build cause (eg. Merge request or Git Push)
- ☐ Build on successful pipeline events

Pending build name for pipeline ?

- ☐ Cancel pending merge request builds on update

Allowed branches
- ⦿ Allow all branches to trigger this job ?
- ○ Filter branches by name ?
- ○ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token ?

~~[secret token]~~

[Generate]

SecretToken을 생성하고 gitlab에 설정해준다. Webhooks 페이지에 들어가 사용할 git clone 주소를 입력하고 밑에 secret Token을 적어주고 저장한다.

그리고 Filter branches by name을 체크하여 Include에 사용할 브랜치인 develop을 적어준다.

develop에만 push가 일어나야 build하게 하는 설정

6. 빌드 환경 설정

빌드 환경에서 Delete workspace before build starts와 Privide Node & npm bin/ folder to PATH를 체크하고 노드를 설정해준다.

7. 마지막으로 Build Steps를 설정해준다.

Invoke Gradel script를 추가 Wrapper location에는 backend 디렉토리 이름을 작성한다.

고급을 눌러 BuildFile에 build.gradle이 있는 위치를 적어준다.

Root Build script ?

Build File ?

Specify Gradle build file to run. Also, **some environment variables are available to the build script**

`[          ]/build.gradle`

☐ Force GRADLE_USER_HOME to use workspace ?

backend 빌드 script

```
if [ $( docker ps -a | grep carum | wc -l ) -gt 0 ]; then
  docker rm -f carum
fi
# application.properties 와 pem키는 git이 아닌 로컬에서 복사해준다.
docker cp jenkins:/var/jenkins_home/workspace/application.properties backend/src/main/resources
docker cp jenkins:/var/jenkins_home/workspace/keystore.p12 backend/src/main/resources

docker build -t carum:latest .
# 외부 이미지와, 음악을 사용하기 위해 볼륨을 잡아 container를 실행해준다.
docker run -it -e TZ=Asia/Seoul --name carum -d -p 8080:8080 -v /home/ubuntu/data/carum/image:/var/carum/image -v /home/ubuntu/data/car
```

frontend 빌드 script

```
cd frontend/carum
# 프론트 환경변수를 저장한 env 파일은 로컬에서 가져온다.
docker cp jenkins:/var/jenkins_home/workspace/.env .

npm install --force
npm run build
# build파일을 nginx에 복사하고 nginx 재시작
docker cp build carum-nginx:/var/www/
docker restart carum-nginx
```

4. docker-compose 작성

```
version: "3.7"
services:
    nginx:
        container_name: carum-nginx
        image: nginx
        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: carum-certbot
        image: certbot/certbot
        volumes:
            - ./data/certbot/conf:/etc/letsencrypt
            - ./data/certbot/www:/var/www/certbot
    mysql:
        container_name: carum-mysql
        image: mysql:5.7
        ports:
            - 3306:3306
        volumes:
            - ./data/mysql:/var/lib/mysql
        environment:
            - MYSQL_ROOT_PASSWORD=dbslxlzkfnaA101
            - MYSQL_DATABASE=carum
```

```
        command:
            - --character-set-server=utf8
            - --collation-server=utf8_general_ci
        restart: always
    redis:
        container_name: carum-redis
        image: redis
        ports:
            - "6379:6379"
        volumes:
            - ./data/redis:/data
        restart: on-failure
```

서버에서 docker-compose up을 해주고 마무리.

Spring boot 설정파일

- application.properties

```
# server.port=8080

# https 적용 Keystore
server.ssl.enabled=true
server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=YOUR_SSL_PASSWORD

# DB 설정
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.url=YOUR_DB_URL
spring.datasource.username=root
spring.datasource.password=YOUR_DB_PASSWORD
spring.profiles.include=local
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.open-in-view=false
spring.jpa.defer-datasource-initialization=true

spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
spring.jpa.database=mysql
spring.jpa.hibernate.ddl-auto=update

# redis 설정
spring.redis.host= k7a101.p.ssafy.io
spring.redis.port= 6379

# 이미지 설정
file.upload.dir=/var/carum/image/
file.upload.url=https://k7a101.p.ssafy.io/api/image/
music.upload.dir=/var/carum/music/
music.upload.url=https://k7a101.p.ssafy.io/api/music/file/

# properties value 설정
jwt.token.secret=YOUR_JWT_SECRET
jwt.token.time.expire = 1440
jwt.token.time.refresh = 10080

room.template.base = 1
room.template.list = 1,2,3,4
```