**포팅 매뉴얼**



**공통 프로젝트**

**<세치혀>**

**팀원 : 김영범,홍성영,정다은,양희경,김혜준,이종인**

**1. 프로젝트 기술 스택**

FrontEnd

- HTML5

- CSS3

- TypeScript(ES6)

- Vue3

- Node.js 16.16.0

BackEnd

- Java11

- SpringBoot Gradle 2.7.2

- MySQL 8.0.28

IDE

- VS Code 1.69

- InteliJ 2022.1.3

Server

- AWS EC2

- Ubuntu 20.04 LTS

- Git

- Openvidu on promises

- Docker

- NginX

**2.빌드 방법 및 배포 방법**

**1) 프로젝트 폴더 가져오기**

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 git에 올린 최종파일들을 Clone 해온다



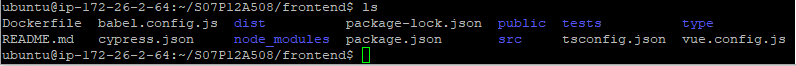
만약 이미 Clone 해놓은 폴더가 있다면 해당 폴더에서 git pull

**2) FrontEnd 빌드 & 배포**



해당 frontEnd프로젝트에서

$ npm run build



ls 명령어로 dist 폴더가 생성됐는지 확인

해당 폴더 내의 index.html이 있는지 확인하고 그 위치로 nginx 의 설정이 제대로 되어 있는지 확인(NginX의 설정은 Nginx 설정 쪽에서 확인 가능)

**3) BackEnd 빌드 & 배포**

backEnd 프로젝트 내부에 들어오면 gradlew가 존재하는 것을 볼 수 있는데

이 때 이 파일의 권한이 없다고 뜨면(초록색이 아니면)

$ chmod 777 gradlew

명령어를 입력하여서 권한 부여 후

$ ./gradlew build

빌드 성공한 것이 확인되면

$ cd build/libs/

로 들어가서 jar 파일이 생성된 것을 확인

**Docker로 배포**

다시 backend폴더로 돌아온 후 DockerFile를 설정

**DockerFile**

FROM openjdk:11  
  
# 도커 컨테이너 외부에 노출할 포트 지정  
EXPOSE 8080  
  
# 도커 내에서 사용할 변수의 값 설정 , 아무튼 그 뒤에 jar 파일 명  
ARG JAR\_FILE=build/libs/backend-0.0.1-SNAPSHOT.jar  
  
# JAR\_FILE에 해당하는 파일 app.jar  
COPY ${JAR\_FILE} app.jar  
  
# CMD와 유사하나, 디폴트로 수행될 명령은 ENTRYPOINT를 사용해 지정  
ENTRYPOINT ["java","-jar","/app.jar"]  
  
# 도커 timezone 변경  
ENV TZ=Asia/Seoul  
RUN apt-get install -y tzdata

**\* 주의사항!!!**

Docker로 8080 포트를 사용해서 배포할 예정인데 이미 그 Port를 사용하거나 하는 전 프로젝트가 있으면 안된다.

$ docker ps -a

명령어를 사용해서 전 프로젝트가 실행되거나 혹은 8080포트를 사용하는 다른 프로젝트가 있는지 확인 후 필요없다면 삭제하거나 아니면 포트 번호를 옮겨주는 등의 작업을 해야한다.

도커 컨테이너 정지 후 삭제

$ docker stop <docker container name>  
$ docker rm <docker container name>

도커 이미지 빌드

$ docker build -t backend .



도커 이미지 확인

$ docker images



도커 컨테이너 생성 후 실행

$ docker run -d -p 8080:8080 --name backend\_container -v /home/ubuntu/resources:/home/ubuntu/resources backend



**3. MySQL 워크벤치 사용**

Aws에 설치한 Docker로 설치한 DB에 매번 Bash로 접속하여 실행하는 건 여러모로 어려움이 있어 워크벤치를 통한 접속

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

위 빨간 부분을 맞는 정보를 기입하여 test Connection하여 성공 시 사용

**4. NginX Default 설정**

**/etc/nginx/sites-enabled/**

**default**

server {  
 listen 80 default\_server;  
 listen [::]:80 default\_server;  
  
 server\_name i7a508.p.ssafy.io;  
 client\_max\_body\_size 500M;  
 return 301 https://$server\_name$request\_uri;  
}  
server {  
  
 listen 443 ssl;  
 listen [::]:443 ssl;  
  
 server\_name i7a508.p.ssafy.io;  
   
   
 ssl\_certificate /etc/letsencrypt/live/i7a508.p.ssafy.io/fullchain.pem; # managed by Certbot  
 ssl\_certificate\_key /etc/letsencrypt/live/i7a508.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  
  
   
  
 location / {  
 root /home/ubuntu/S07P12A508/frontend/dist;  
 index index.html index.htm;  
 try\_files $uri $uri/ /index.html;  
 }  
  
 location /api {  
 proxy\_pass http://localhost:8080/api;  
 proxy\_redirect off;  
 charset utf-8;  
  
 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;  
  
 proxy\_buffer\_size 128k;  
 proxy\_buffers 4 256k;  
 proxy\_busy\_buffers\_size 256k;  
 }  
  
 location /images {  
 root /home/ubuntu/resources;  
 }  
}

**5.EC2 설정**

**1.apt update**

$ sudo apt update

**2.docker 설치**

$ sudo wget -qO- http://get.docker.com/ | sh

**3.node.js 설치**

$ sudo apt install nodejs

**4.npm 설치**

$ sudo apt install npm

**5.MySql 을 docker에 설치**

$ docker pull mysql:8.0.28

**-MySql 이미지를 실행(password엔 설정할 root계정의 비밀번호 입력)**

$ docker run --name mysql-container -e MYSQL\_ROOT\_PASSWORD=<password> -d -p 3306:3306 mysql:latest

**6.Openvidu on Premises 방식 설치**

Root 권한 얻기

$ sudo su

/opt 이동

$ cd /opt

Openvidu 설치

$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install\_openvidu\_latest.sh | bash

설치된 경로로 이동

$ cd openvidu

설정파일 실행

$ vi .env

**.env**

DOMAIN\_OR\_PUBLIC\_IP=i7a508.p.ssafy.io  
  
OPENVIDU\_SECRET=MY\_SECRET  
  
CERTIFICATE\_TYPE=letsencrypt  
  
LETSENCRYPT\_EMAIL=user@example.com  
  
# HTTP port  
HTTP\_PORT=8442  
  
# HTTPS port(해당 포트를 통해 오픈비두 서버와 연결)  
HTTPS\_PORT=8443

DOMAIN\_OR\_PUBLIC\_IP= 도메인 이름

OPENVIDU\_SECRET=내가 설정할 비밀번호

CERTIFICATE\_TYPE=letsencrypt ( 인증서를 발급받아 사용)

LETSENCRYPT\_EMAIL = 실제 사용가능한 이메일

HTTP\_PORT=사용가능한 HTTP에 쓸 포트 입력

HTTPS\_PORT = 사용가능한 HTTPS에 쓸 포트 입력

\*\*\*처음 HTTP\_PORT와 HTTPS\_PORT는 기존 설정(80, 443)사용-> 처음 인증서를 받아오기 위함!

Openvidu 실행

$ ./openvidu start

Docker ps 명령어로 제대로 돌아가는지 확인

**7.NginX 설치**

Nginx 설치

$ sudo apt install nginx

Nginx 실행

$ sudo service start nginx

Nginx 재실행

$ sudo service restart nginx

**8.NginX에 CertBot-LetsEncrypt 적용(https 처리)**

certBot 다운로드

$ sudo add-apt-repository ppa:certbot/certbot  
$ sudo apt-get install python-certbot-nginx

certBot 을 통해 인증서 발급받으면서 nginx 에 적용

$ sudo certbot --nginx -d <도메인 이름>

\*\*위 명령어 실행 시 이메일 입력은 유효한 이메일을 입력해야 하고 옵션은 필요에 따라 설정해서 설치하면 된다.