

목차

개요

프로젝트 사용도구

환경변수

배포

서비스 시나리오

개요

코로나 판데믹 이 후 대면으로만 이루어지던 다양한 서비스, 행위들이 비대면의 형태로 바뀌었다. 그 대표적인 예로 면접을 들 수 있는 데, 기존의 면접의 형태는 대부분 대면

으로 이루어지는 게 당연하였으나 코로나 판데믹 이 후 화상면접의 형태로 면접이 많이

이루어지게 되었다. 본 프로젝트는 이와 같은 변화에 맞추어 화상면접에 특화된 웹 솔루

션 개발을 개발하여 화상 면접에 알맞은 프로세스를 갖추지 못한 기업이나 채용프로세스 가 구축되지 않은 기업들이 최소한의 고려만으로 성공적인 채용을 이뤄지게 함에 목적이 있다.

1. 프로젝트 사용도구

이슈 관리: Jira

형상 관리: Gitlab

커뮤니케이션: Notion, Mattermost

디자인: Figma

UCC: 모바비

CICD: JENKINS

1. 개발환경

VS Code:

Intellij

JAVA: julu openJdk 8버전

Node.js: 16.19.0

SERVER: AWS AWS EC2 Ubuntu 20.04.3 LTS

Database: MYSQL(Docker)

1. 환경변수

Application.yml [Spring 환경변수]

spring:

web:// database source

resources:

add-mappings: false

datasource:

url: jdbc:mysql://localhost:3306/[DBNAME]?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=true

hikari:

username: // mysql username

password: // mysql database\_password

driver-class-name: com.mysql.cj.jdbc.Driver

jpa: // jpa 환경변수

database-platform: org.hibernate.dialect.MySQL57Dialect // mysql 설정

hibernate:

ddl-auto: update

properties:

hibernate:

format\_sql: true

show\_sql: true

data:

web:

pageable:

one-indexed-parameters: true

redis:

host: // redis address

port: //redis port

password: blackbunny

mvc:

throw-exception-if-no-handler-found: true

pathmatch:

matching-strategy: ant\_path\_matcher

servlet: // 파일 업로드 설정

multipart:

enabled: true

file-size-threshold: 2KB

max-file-size: 500KB

max-request-size: 215MB

mail: // mail function

properties:

mail:

smtp:

starttls:

enable: true

required: true

auth: true

host: smtp.gmail.com

port: 587

username: // mail account

password: // mail account password

jwt:

secret: // jwt secret key

expiration: //jwt 만료 시간

Docker 설정

DockerFile

Backend

FROM [openjdk version]

ARG JAR\_FILE\_PATH=build/libs/\*.jar

COPY ${JAR\_FILE\_PATH} app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

RUN bash -c 'mkdir -pv /etc/{logo,introduce,applicant,evaluator}'

// 이미지 저장 folder 내부 생성

FrontEnd

FROM [node version] as build-stage

RUN npm install -g http-server

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE // port number

CMD [ "http-server", "dist" ]

Jenkins [ DinD Excute Shell]

cd backend

cd focus

chmod +x gradlew

./gradlew build

docker build -t focus\_backend .

docker stop backserver

docker rm backserver

docker run -p 7085:8082 --name backserver --network [network\_name] -e SPRING\_DATASOURCE\_URL=jdbc:mysql://[ipaddress]:3306/focus\_db -v /applicant:/etc/applicant -v /evaluator:/etc/evaluator -v /logo:/etc/logo -v /introduce:/etc/introduce -e SPRING\_DATASOURCE\_HIKARI\_USERNAME=[name] -e SPRING\_DATASOURCE\_HIKARI\_PASSWORD=[password] -e REDIS\_HOST=host.docker.internal -d focus\_backend

cd frontend/vue-focus

docker build -t focus\_front -f ./Dockerfile .

docker stop frontserver

docker rm frontserver

docker run -d -p 9902:8080 --name frontserver --network [network\_name] focus\_front

MYSQL, Redis, Jenkins Docker-compsoe.yml

version: "3"

services:

mysql:

image: mysql

container\_name: mysql

environment:

MYSQL\_DATABASE: [YOUR\_DB\_NAME]

MYSQL\_ROOT\_PASSWORD: [YOUR\_PASSWORD]

MYSQL\_USER: [YOUR\_USERNAME]

MYSQL\_PASSWORD : [YOUR\_PASSWORD]

volumes:

- /mysql:/var/lib/mysql

ports:

- 3306:3306

redis:

container\_name: redis

image: redis

ports:

- 6379:6379

environment:

- REDIS\_PASSWORD=[YOUR\_PASSWROD]

jenkins:

image: jenkins/jenkins:lts

container\_name: jenkins

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- /jenkins\_home:/var/jenkins\_home

ports:

- 9090:8080

privileged: true

user: root

OpenVidu Option [on promise]

version: '3.1'

services:

openvidu-server:

image: openvidu/openvidu-server:2.25.0

restart: on-failure

network\_mode: host

entrypoint: ['/usr/local/bin/entrypoint.sh']

volumes:

- ./coturn:/run/secrets/coturn

- /var/run/docker.sock:/var/run/docker.sock

- ${OPENVIDU\_RECORDING\_PATH}:${OPENVIDU\_RECORDING\_PATH}

- ${OPENVIDU\_RECORDING\_CUSTOM\_LAYOUT}:${OPENVIDU\_RECORDING\_CUSTOM\_LAYOUT}

- ${OPENVIDU\_CDR\_PATH}:${OPENVIDU\_CDR\_PATH}

env\_file:

- .env

environment:

- SERVER\_SSL\_ENABLED=false

- SERVER\_PORT=5443

- KMS\_URIS=["ws://localhost:8888/kurento"]

- COTURN\_IP=${COTURN\_IP:-auto-ipv4}

- COTURN\_PORT=${COTURN\_PORT:-3478}

logging:

options:

max-size: "${DOCKER\_LOGS\_MAX\_SIZE:-100M}"

kms:

image: ${KMS\_IMAGE:-kurento/kurento-media-server:6.18.0}

restart: always

network\_mode: host

ulimits:

core: -1

volumes:

- /opt/openvidu/kms-crashes:/opt/openvidu/kms-crashes

- ${OPENVIDU\_RECORDING\_PATH}:${OPENVIDU\_RECORDING\_PATH}

- /opt/openvidu/kurento-logs:/opt/openvidu/kurento-logs

environment:

- KMS\_MIN\_PORT=40000

- KMS\_MAX\_PORT=57000

- GST\_DEBUG=${KMS\_DOCKER\_ENV\_GST\_DEBUG:-}

- KURENTO\_LOG\_FILE\_SIZE=${KMS\_DOCKER\_ENV\_KURENTO\_LOG\_FILE\_SIZE:-100}

- KURENTO\_LOGS\_PATH=/opt/openvidu/kurento-logs

logging:

options:

max-size: "${DOCKER\_LOGS\_MAX\_SIZE:-100M}"

coturn:

image: openvidu/openvidu-coturn:2.25.0

restart: on-failure

ports:

- "${COTURN\_PORT:-3478}:${COTURN\_PORT:-3478}/tcp"

- "${COTURN\_PORT:-3478}:${COTURN\_PORT:-3478}/udp"

env\_file:

- .env

volumes:

- ./coturn:/run/secrets/coturn

command:

- --log-file=stdout

- --listening-port=${COTURN\_PORT:-3478}

- --fingerprint

- --min-port=${COTURN\_MIN\_PORT:-57001}

- --max-port=${COTURN\_MAX\_PORT:-65535}

- --realm=openvidu

- --verbose

- --use-auth-secret

- --static-auth-secret=$${COTURN\_SHARED\_SECRET\_KEY}

logging:

options:

max-size: "${DOCKER\_LOGS\_MAX\_SIZE:-100M}"

nginx:

image: openvidu/openvidu-proxy:2.25.0

restart: always

network\_mode: host

volumes:

- ./certificates:/etc/letsencrypt

- ./owncert:/owncert

- ./custom-nginx-vhosts:/etc/nginx/vhost.d/

- ./custom-nginx-locations:/custom-nginx-locations

- ${OPENVIDU\_RECORDING\_CUSTOM\_LAYOUT}:/opt/openvidu/custom-layout

- ./custom-nginx.conf:/custom-nginx/custom-nginx.conf

- ./nginx.conf:/etc/nginx/nginx.conf

environment:

- DOMAIN\_OR\_PUBLIC\_IP=${DOMAIN\_OR\_PUBLIC\_IP}

- CERTIFICATE\_TYPE=${CERTIFICATE\_TYPE}

- LETSENCRYPT\_EMAIL=${LETSENCRYPT\_EMAIL}

- PROXY\_HTTP\_PORT=${HTTP\_PORT:-}

- PROXY\_HTTPS\_PORT=${HTTPS\_PORT:-}

- PROXY\_HTTPS\_PROTOCOLS=${HTTPS\_PROTOCOLS:-}

- PROXY\_HTTPS\_CIPHERS=${HTTPS\_CIPHERS:-}

- PROXY\_HTTPS\_HSTS=${HTTPS\_HSTS:-}

- ALLOWED\_ACCESS\_TO\_DASHBOARD=${ALLOWED\_ACCESS\_TO\_DASHBOARD:-}

- ALLOWED\_ACCESS\_TO\_RESTAPI=${ALLOWED\_ACCESS\_TO\_RESTAPI:-}

- PROXY\_MODE=CE

- WITH\_APP=true

- SUPPORT\_DEPRECATED\_API=${SUPPORT\_DEPRECATED\_API:-false}

- REDIRECT\_WWW=${REDIRECT\_WWW:-false}

- WORKER\_CONNECTIONS=${WORKER\_CONNECTIONS:-10240}

- PUBLIC\_IP=${PROXY\_PUBLIC\_IP:-auto-ipv4}

logging:

options:

max-size: "${DOCKER\_LOGS\_MAX\_SIZE:-100M}"

Custom-nginx-conf

version: '3.1'

upstream openviduserver {

server localhost:5443;

}

upstream backend {

server localhost:7085;

}

upstream frontend{

server localhost:9902;

}

server {

listen 80;

listen [::]:80;

server\_name [your\_hostname]

# Redirect to https

location / {

rewrite ^(.\*)[your address]:443$1 permanent;

}

# letsencrypt

location /.well-known/acme-challenge/ {

root /var/www/certbot;

}

location /nginx\_status {

stub\_status;

allow 127.0.0.1; #only allow requests from localhost

deny all; #deny all other hosts

}

}

server {

listen 443 ssl;

listen [::]:443 ssl;

server\_name [your\_hostname];

# SSL Config

ssl\_certificate /etc/letsencrypt/live/i8a106.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/i8a106.p.ssafy.io/privkey.pem;

ssl\_trusted\_certificate /etc/letsencrypt/live/i8a106.p.ssafy.io/fullchain.pem;

ssl\_session\_cache shared:SSL:50m;

ssl\_session\_timeout 5m;

ssl\_stapling on;

ssl\_stapling\_verify on;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384";

ssl\_prefer\_server\_ciphers off;

add\_header Strict-Transport-Security "max-age=63072000" always;

# Proxy

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\_set\_header X-Forwarded-Proto https;

proxy\_headers\_hash\_bucket\_size 512;

proxy\_redirect off;

# Websockets

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

# Your App

location / {

proxy\_pass http://frontend; # Openvidu call by default

}

location /api {

proxy\_pass http://backend;

}

########################

# OpenVidu Locations #

########################

#################################

# Common rules #

#################################

# Dashboard rule

location /dashboard {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

# Websocket rule

location ~ /openvidu$ {

proxy\_pass http://openviduserver;

}

#################################

# New API #

#################################

location /openvidu/layouts {

rewrite ^/openvidu/layouts/(.\*)$ /custom-layout/$1 break;

root /opt/openvidu;

}

location /openvidu/recordings {

proxy\_pass http://openviduserver;

}

location /openvidu/api {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

location /openvidu/info {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

location /openvidu/accept-certificate {

proxy\_pass http://openviduserver;

}

location /openvidu/cdr {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

#################################

# LetsEncrypt #

#################################

location /.well-known/acme-challenge {

root /var/www/certbot;

try\_files $uri $uri/ =404;

}

}

3. 배포

1. Docker를 사용하여 서버를 올리기 때문에 서버 위에 우분투를 설치

합니다.

2. Docker가 완료되면 2챕터에 기재되어 있는 MYSQL,JEKINS,REDIS의

설정이 입력된 컴포즈 파일을 실행시킵니다.

3. Jenkins 내부에 Docker를 설치하여 DinD 환경을 만듭니다.

4. DinD 환경이 구성되면 Jenkin gitlab plugin, docker plugin을 다운받습니다.

그리고 Gitlab에서 Webhook을 연결시켜줍니다. 이 후 frontend, backend를

자동 빌드해주고 docker run 명령어를 통하여 실행시킵니다.

1. 서비스 시나리오

1. 기업관리자 계정 할당 - 서비스 관리자(root admin) 계정으로 로그인 - 계정관리 메뉴로 들어간 후 우측 상단 계정 생성 클릭 - 정보 입력 후 계정 생성 버튼 클릭

2. 평가자 계정 할당 - 기업관리자 계정으로 로그인 - 평가자 관리 메뉴로 들어간 후 우측 상단 평가자 계정 생성 클릭 - 정보 입력 후 계정 생성 버튼 클릭 - 리스트에서 생성한 계정을 볼 수 있으며, 계정 할당 버튼을 통해 아이디, 비밀번호 자동 생성 됨 - 생성된 ID와 비밀번호는 입력한 개인 이메일로 발송되어 확인할 수 있음

3. 면접 전형 등록 - 프로세스 관리 메뉴로 들어간 후 전형 기간 입력, 전형명, 총 전형차수 입력 후 프로세스 생성 - 전형 리스트 중 원하는 전형을 클릭하고, 세부 전형을 등록 - 하단의 전형 추가 버튼으로 세부 전형의 시작일, 종료일, 전형명, 전형 차수, 평가지 등록 후 저장

4. 면접 전형에 지원자 배정 - 지원자 관리 메뉴로 들어간 후 지원자 개인 정보 입력 후 계정 생성 버튼 클릭 - 지원자 리스트 조회에서 생성한 지원자의 계정 할당 가능 - 할당된 아이디와 비밀번호는 지원자 개인 이메일로 발송되어 확인할 수 있음

5. 면접 일정 등록 - 일정을 등록하고 싶은 전형을 클릭 후 차수 별 면접을 생성 - 면접명, 일시, 시작, 종료시간, 평가자와 지원자를 지정한 후 면접 생성 버튼 클릭

6. 면접 진행 [평가자] - 평가자 계정으로 로그인 - 입장 버튼으로 면접실 입장 - 카메라 on/off, 마이크 on/off 가능 - 우측 지원자 이름을 클릭하여 평가지를 볼 수 있으며, 작성 후 하단 제출 버튼을 눌러 평가 반영 [지원자] - 로그인 후 입장 버튼으로 면접실 입장 - 카메라 on/off, 마이크 on/off 가능

7. 지원자 합격/불합격 처리 - 합격하고 싶은 지원자를 체크 박스로 처리한 후 우측 상단 전형 종료 클릭