

포팅 매뉴얼

미리보기

1. 개발 환경 및 외부 설정

- 1) 개발 환경
- 2) 외부 서비스

2. 서비스 빌드 및 배포

- 1) Docker 및 docker-compose 설치
- 2) openvidu 설치
- 3) openvidu 중계서버 설정
- 4) openvidu 설정
- 5) 서비스 docker-compose.yml 작성
- 6) Redis 설정
- 7) elasticsearch 설정
- 8) Jenkins 설정
- 9) MySQL 덤프파일 불러오기
- 10) Jenkins 빌드 및 배포

3. 배포 시 특이 사항

※ openvidu-server 를 종료시킨 후 재실행하게 되면 nginx의 reverse-proxy 설정이 초기화 됩니다.

4. 프로젝트 설정 파일 정리

※ 해당 내용은 2. 서비스 빌드 및 배포 에서 함께 다루고 있습니다.

1. 개발 환경 및 외부 설정

1) 개발 환경

- IDE : IntelliJ 2023.01 / VSCode / MySQL Workbench / Android Studio 2024.01

- **Database** : MySQL 8.0.39 / Redis
- **jdk** : openjdk-17-Temurin
- **JS Runtime** : Node.js 20.16
- **Frontend** : React.js / Vite / Redux / Zustand / Axios / StyledCompoents / eslint (Airbnb)
- **Backend** : Spring Boot 3.3.2 / spring-data-jpa / Spring Security / jjwt / lombok
- **WebRTC** : Openvidu 2.23.0
- **Infra** : docker-compose / Jenkins / Nginx
- **Tools** : Jira / Figma / Notion

2) 외부 서비스

- Google STT API
- 카카오맵/모빌리티 API
- OPEN API (전국의료기관 응급실 현황)
- 데이터 (소방청 시도 소방서 현황 / 의약품 제품허가 목록)
- Firebase Cloud Messaging

2. 서비스 빌드 및 배포 매뉴얼

※ 해당 내용은 Linux (Ubuntu 20.04)를 기준으로 작성되었습니다.

1) Docker 및 docker-compose 설치

```
# Docker 설치
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
sudo add-apt-repository "deb [arch=amd64] https://download.do
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io

# docker-compose 설치
sudo curl -L "https://github.com/docker/compose/releases/down
```

```
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-comp
```

2) openvidu 설치

```
sudo su
cd /opt
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install.sh
cd openvidu
vi .env # openvidu 환경변수 설정 후 저장
```

.env 파일 내의 환경변수 설정

```
DOMAIN_OR_PUBLIC_IP=i11b305
CERTIFICATE_TYPE=letsencrypt
OPENVIDU_SECRET=YSnUC1WgXb8D4NQjSQGvnTX07b0Epy
LETSencrypt_EMAIL=hosan_1@naver.com
```

실행 테스트

```
./openvidu start
```

3) openvidu 중계서버 설정

```
mkdir node
cd node
vi server.js # server.js 내용 작성 후 저장
vi package.json # package.json 내용 작성 후 저장
cd .. # /opt/openvidu 디렉토리로 이동
vi Dockerfile # Dockerfile 내용 작성 후 저장
```

server.js 내용

```
require("dotenv").config();
var express = require("express");
var bodyParser = require("body-parser");
var http = require("http");
```

```

var cors = require("cors");
var { OpenVidu } = require("openvidu-node-client");

var app = express();

var SERVER_PORT = 5442;
var OPENVIDU_URL = "http://localhost:4443";
var OPENVIDU_SECRET = "YSnUC1WgXb8D4NQjSQGvnTX07b0Epy";

// Enable CORS support
app.use(cors({ origin: "*" }));

var server = http.createServer(app);
var openvidu = new OpenVidu(OPENVIDU_URL, OPENVIDU_SECRET);

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

server.listen(SERVER_PORT, () => {
  console.log("Application started on port:", SERVER_PORT);
  console.warn("Application server connecting to OpenVidu at");
});

app.post("/api/sessions", async (req, res) => {
  try {
    await openvidu.fetch();
    var session = await openvidu.createSession(req.body);
    res.status(200).send(session.sessionId);
    console.log("Create session.");
  } catch (error) {
    console.error("Error creating session:", error);
    res.status(500).send(error.message);
  }
});

app.post("/api/sessions/:sessionId/connections", async (req,
  try {

```

```

    await openvidu.fetch();
    var session = openvidu.activeSessions.find(
      (s) => s.sessionId === req.params.sessionId
    );
    if (!session) {
      res.status(404).send("현재 활성화된 Session이 없습니다.");
    } else {
      var connection = await session.createConnection(req.body);
      res.status(200).send(connection.token);
      console.log("Connection");
    }
  } catch (error) {
    console.error("Error creating connection:", error);
    res.status(500).send(error.message);
  }
});

```

// activeSessions 가 0이면 활성화된 session이 없다고 반환
 // activeSessions 하나하나 보면서 active connections 이 없으면 con

```

app.get("/api/sessions/rooms", async (req, res) => {

  try{

    // 서버의 최신 session 정보를 가져온다.
    await openvidu.fetch();

    console.log(
      openvidu.activeSessions[0].connections[0])
    if (openvidu.activeSessions.length === 0) {
      return res.status(404).send("현재 활성화된 Session이 없습니다.");
    }
    //
    // 활성화된 세션 중 activeConnections가 1인 세션을 찾음
    let session = null;
    for (let s of openvidu.activeSessions) {
      // session의 최신 정보 가져오기
      await s.fetch();
    }
  }
}

```

```

    if (s.activeConnections.length === 1) {
        session = s;
        break;
    }
}

if (session) {
    let resData = {
        sessionId: session.sessionId
    }
    return res.status(200).send(resData);
} else {
    return res.status(404).send("현재 연결 가능한 Connection이 없습니다.");
}
} catch (error) {
    console.error("Error fetching sessions:", error);
    return res.status(500).send("서버 에러가 발생했습니다.");
}
})

process.on("uncaughtException", (err) => {
    console.error("Uncaught Exception:", err);
});

```

package.json 내용

```

{
  "name": "server",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

```

```
"dependencies": {
  "body-parser": "^1.20.2",
  "cors": "^2.8.5",
  "dotenv": "^16.4.5",
  "express": "^4.19.2",
  "openvidu-node-client": "^2.23.0"
}
```

Dockerfile 내용

```
# Use the official Node.js image as a base
FROM node:20

# Set the working directory
WORKDIR ./node

# Copy package.json and package-lock.json
COPY ./node/package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application code
COPY ./node .

# Expose the port that the server will run on
EXPOSE 5442

# Start the server
CMD ["node", "server.js"]
```

4) openvidu 설정

```
vi docker-compose.yml # /opt/openvidu/docker-compose.yml 내용
```

/opt/openvidu/docker-compose.yml 내용

```
version: '3.1'
```

```
services:
```

```
  openvidu-server:
```

```
    container_name: openvidu-server
```

```
    image: openvidu/openvidu-server:2.23.0
```

```
    network_mode: host
```

```
    restart: on-failure
```

```
    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=4443
```

```
      - KMS_URI=[ "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:
```

```
    container_name: kms
```

```
    image: ${KMS_IMAGE:-kurento/kurento-media-server:6.18}
```

```
    network_mode: host
```

```
    restart: always
```

```
    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_
  - KURENTO_LOGS_PATH=/opt/openvidu/kurento-logs
logging:
  options:
    max-size: "${DOCKER_LOGS_MAX_SIZE:-100M}"
coturn:
  container_name: coturn
  image: openvidu/openvidu-coturn:2.23.0
  restart: on-failure
  ports:
    - "${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}"
  network_mode: host
nginx:
  container_name: openvidu-proxy
  image: openvidu/openvidu-proxy:2.23.0
  restart: always

```

```

network_mode: host
volumes:
  - ./certificates:/etc/letsencrypt
  - ./owncert:/owncert
  - ./custom-nginx-vhosts:/etc/nginx/vhost.d/
  - ./custom-nginx-locations:/custom-nginx-location
  - ${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:/opt/openvidu
environment:
  - DOMAIN_OR_PUBLIC_IP=${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:-}
  - 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}"
server:
  container_name: openvidu-node
  network_mode: host
  build:
    context: .
    dockerfile: Dockerfile
  ports:
    - "5442:5442"
  environment:
    - OPENVIDU_URL=http://localhost:4443
    - OPENVIDU_SECRET=YsnUC1WgXb8D4NQjSQGvnTX07b0Epy

```

./openvidu 파일의 내용 변경

```
# ...

case $1 in
  start)
    docker-compose up --build -d
    docker-compose logs -f --tail 10 server
    ;;

  stop)
    docker-compose stop openvidu-node
    ;;

  allstop)
    docker-compose down
    ;;

  restart)
    docker-compose stop server
    docker-compose up --build -d server
    docker-compose logs -f --tail 10 server
    ;;

# ...
```

5) 서비스 docker-compose.yml 작성

```
cd /home/ubuntu
mkdir dockercompose
cd dockercompose
vi docker-compose.yml
```

/home/ubuntu/dockercompose/docker-compose.yml 내용

```
version: '3'
services:
  setup:
```

```

profiles:
  - setup
build:
  context: setup/
  args:
    ELASTIC_VERSION: ${ELASTIC_VERSION}
init: true
volumes:
  - ./setup/entrypoint.sh:/entrypoint.sh:ro,Z
  - ./setup/lib.sh:/lib.sh:ro,Z
  - ./setup/roles:/roles:ro,Z
environment:
  ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
  LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSWORD:-}
  KIBANA_SYSTEM_PASSWORD: ${KIBANA_SYSTEM_PASSWORD:-}
  METRICBEAT_INTERNAL_PASSWORD: ${METRICBEAT_INTERNAL_PASSWORD:-}
  FILEBEAT_INTERNAL_PASSWORD: ${FILEBEAT_INTERNAL_PASSWORD:-}
  HEARTBEAT_INTERNAL_PASSWORD: ${HEARTBEAT_INTERNAL_PASSWORD:-}
  MONITORING_INTERNAL_PASSWORD: ${MONITORING_INTERNAL_PASSWORD:-}
  BEATS_SYSTEM_PASSWORD: ${BEATS_SYSTEM_PASSWORD:-}
depends_on:
  - elasticsearch
  - logstash
certbot:
  image: certbot/certbot
  restart: unless-stopped
  volumes:
    - ./data/certbot/conf:/etc/letsencrypt
    - ./data/certbot/www:/var/www/certbot
  entrypoint: "/bin/sh -c 'trap exit TERM; while :; do cert
jenkins:
  container_name: jenkins
  image: jenkins/jenkins:lts-jdk17
  restart: unless-stopped
  environment:
    JENKINS_OPTS: "--prefix=/jenkins"
  ports:
    - "8080:8080"

```

```

    - "50000:50000"
  volumes:
    - ./jenkins-data:/var/jenkins_home
  user: root
nginxapp:
  container_name: app
  image: nginx:latest
  volumes:
    - ./frontend/app/default.conf:/etc/nginx/conf.d/default
    - ./frontend/app/index.html:/usr/share/nginx/html/index
    - ./frontend/app/assets/:/usr/share/nginx/html/assets/
  restart: always
  ports:
    - "3001:3001"
nginx:
  container_name: web
  image: nginx:latest
  volumes:
    - ./frontend/rescu/default.conf:/etc/nginx/conf.d/default
    - ./frontend/rescu/index.html:/usr/share/nginx/html/index
    - ./frontend/rescu/assets/:/usr/share/nginx/html/assets
  restart: always
  ports:
    - "3002:3002"
mysql:
  container_name: mysql
  image: mysql:8.0.39
  environment:
    MYSQL_ROOT_PASSWORD: b305!@Jukah4kV4N
    MYSQL_DATABASE: smru
    MYSQL_USER: ssafyb305
    MYSQL_PASSWORD: b305!@z6bJOz9JBf
    TZ: 'Asia/Seoul'
  ports:
    - "3306:3306"
  volumes:
    - ./mysql-data:/var/lib/mysql
api:

```

```

container_name: api
restart: on-failure
build:
  context: ./
  dockerfile: Dockerfile
ports:
  - "5000:5000"
environment:
  SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/smru?use
  SPRING_DATASOURCE_USERNAME: "ssafyb305"
  SPRING_DATASOURCE_PASSWORD: "b305!@z6bJ0z9JBf"
  SPRING_REDIS_HOST: "redis"
  SPRING_REDIS_PORT: 6379
volumes:
  - /home/ubuntu/dockercompose
depends_on:
  - mysql
  - elasticsearch
  - redis
redis:
  container_name: redis
  image: redis:latest
  ports:
    - "6379:6379"
  command: redis-server /usr/local/etc/redis/redis.conf
  volumes:
    - ./redis/data:/data
    - ./redis/redis.conf:/usr/local/etc/redis/redis.conf
  restart: always
elasticsearch:
  container_name: elasticsearch
  build:
    context: elasticsearch/
    args:
      ELASTIC_VERSION: ${ELASTIC_VERSION}
  volumes:
    - ./elasticsearch/config/elasticsearch.yml:/usr/share/e
    - elasticsearch:/usr/share/elasticsearch/data:Z

```

```

ports:
  - 9200:9200
  - 9300:9300
environment:
  node.name: elasticsearch
  ES_JAVA_OPTS: -Xms512m -Xmx512m
  # Bootstrap password.
  # Used to initialize the keystore during the initial start.
  # Elasticsearch. Ignored on subsequent runs.
  ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
  # Use single node discovery in order to disable product.
  # see: https://www.elastic.co/guide/en/elasticsearch/re
  discovery.type: single-node
restart: unless-stopped
logstash:
  container_name: logstash
  build:
    context: logstash/
    args:
      ELASTIC_VERSION: ${ELASTIC_VERSION}
  volumes:
    - ./logstash/config/logstash.yml:/usr/share/logstash/co
    - ./logstash/pipeline:/usr/share/logstash/pipeline:ro,Z
    - ./logstash/input:/usr/share/logstash/input
  ports:
    - 5044:5044
    - 50000:50000/tcp
    - 50000:50000/udp
    - 9600:9600
  environment:
    LS_JAVA_OPTS: -Xms256m -Xmx256m
    LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSWOR
  depends_on:
    - elasticsearch
  restart: unless-stopped
volumes:
  elasticsearch:
    driver: local

```

SpringBoot Dockerfile 작성

```
vi Dockerfile
```

```
FROM openjdk:17-jdk
```

```
# 지정된 JAR 파일을 이미지 내의 app.jar로 복사합니다.
```

```
COPY ./spring/*.jar app.jar
```

```
# prod 프로필로 Spring Boot 애플리케이션을 실행합니다.
```

```
ENTRYPOINT ["java", "-jar", "app.jar", "--spring.profiles.act
```

웹서버 설정을 위한 세팅

```
mkdir frontend
```

```
cd frontend
```

```
mkdir app
```

```
mkdir rescu
```

```
vi app/default.conf # app 설정 작성
```

```
vi rescu/default.conf # web 설정 작성
```

app default.conf 내용

```
server {  
    listen 3001; # App  
    root /usr/share/nginx/html;  
    index index.html index.htm;  
    try_files $uri /index.html;  
}
```

web default.conf 내용

```
server {  
    listen 3002; # Web  
    root /usr/share/nginx/html;  
    index index.html index.htm;
```



```
    try_files $uri /index.html;
}
```

spring-deploy.sh 생성

```
cd /home/ubuntu/dockercompose
vi spring-deploy.sh
```

spring-deploy.sh 내용

```
#!/bin/bash

echo "====Build SpringBoot Dockerfile===="
sudo docker build --no-cache -f /home/ubuntu/dockercompose/Dockerfile

echo "====Restart API Server===="
sudo docker-compose -f /home/ubuntu/dockercompose/docker-compose.yml up
sudo docker-compose -f /home/ubuntu/dockercompose/docker-compose.yml restart
sudo docker-compose -f /home/ubuntu/dockercompose/docker-compose.yml down
```

6) Redis 설정

redis 디렉토리 생성 후 redis.conf 파일 작성

```
mkdir redis
vi redis/redis.conf
```

redis.conf 내용은 [이곳](#)에서 참조할 수 있다.

redis.conf 내용 중 아래 내용을 수정한다.

```
#bind 127.0.0.1 -:::1
bind 0.0.0.0
```

7) elasticsearch 설정

※ Clone 받은 폴더 중 es 폴더안의 데이터를 우분투의
/home/ubuntu/dockercompose 로 이동시킨 후 작업을 진행한다.

docker-compose.yml 실행

```
cd /home/ubuntu/dockercompose
docker-compose up -d
```

openvidu 실행 후 reverse-proxy 설정

```
cd /opt/openvidu
./openvidu
docker exec -it openvidu-proxy bash
vi /etc/nginx/conf.d/default.conf # default.conf 내용 수정
nginx -s reload # nginx reload
exit # nginx 컨테이너 접속 종료
```

default.conf 내용

```
upstream yourapp {
    server localhost:5442;
}

upstream openviduserver {
    server localhost:4443;
}

server {
    listen 80;
    listen [::]:80;
    server_name i11b305.p.ssafy.io;

    # Redirect to https
    location / {
        rewrite ^(.*) https://i11b305.p.ssafy.io: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 local
        deny all;                #deny all other hosts
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name i11b305.p.ssafy.io;

    # SSL Config
    ssl_certificate      /etc/letsencrypt/live/i11b305.p.s
    ssl_certificate_key  /etc/letsencrypt/live/i11b305.p.s
    ssl_trusted_certificate /etc/letsencrypt/live/i11b305.p.s

    ssl_session_cache shared:SSL:50m;
    ssl_session_timeout 5m;
    ssl_stapling on;
    ssl_stapling_verify on;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AE

    ssl_prefer_server_ciphers on;

    # Proxy
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_f
    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";

location / {
    return 301 https://$host/rescu;
}

#####
# 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 bro
    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;
}
location /app/ {
    proxy_pass http://localhost:3001/;
    proxy_redirect default;

    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;
}

location /rescu/ {
    proxy_pass http://localhost:3002/;
    proxy_redirect default;

    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;
}

location /api/ {
    proxy_pass http://localhost:5000/api/;
    proxy_redirect default;

    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;
}

location /webrtc/ {
    proxy_pass http://localhost:5442/;
    proxy_redirect default;

    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;
}

location /elasticsearch/ {
    proxy_pass http://localhost:9200/;
    proxy_redirect default;
}

```

```

        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;
    }
    # Jenkins
    location /jenkins {
        proxy_pass http://localhost:8080/jenkins;
        proxy_redirect default;

        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;
    }
}

```

elasticsearch 인덱스 생성

Postman 을 사용해 아래와 같은 방식으로 요청한다.

의약품 정보 인덱스 생성

```

// PUT 방식으로 https://i11b305.p.ssafy.io/elasticsearch/medici
// 아래는 Body 에 들어갈 데이터이다.
{
  "analysis": {
    "tokenizer": {
      "my_ngram_tokenizer": {
        "type": "edge_ngram",
        "min_gram": 1,
        "max_gram": 33
      }
    },
    "filter": {
      "stopwords": {
        "type": "stop",
        "stopwords": [ " " ]
      }
    }
  }
}

```

```

    }
  },
  "analyzer": {
    "my_ngram_analyzer": {
      "type": "custom",
      "tokenizer": "my_ngram_tokenizer",
      "filter": ["trim", "stopwords"]
    }
  }
},
"properties": {
  "medicine_id": {
    "type": "integer"
  },
  "medicine_name": {
    "type": "text",
    "analyzer": "standard",
    "search_analyzer": "standard",
    "fields": {
      "ngram": {
        "type": "text",
        "analyzer": "my_ngram_analyzer",
        "search_analyzer": "my_ngram_analyzer"
      }
    }
  }
}
}

```

지병 데이터 인덱스 생성

```

// PUT 방식으로 https://i11b305.p.ssafy.io/elasticsearch/cdinfo
// 아래는 Body 에 들어갈 데이터이다.
{
  {
    "analysis": {
      "tokenizer": {
        "my_ngram_tokenizer": {

```



```

        "type": "edge_ngram",
        "min_gram": 1,
        "max_gram": 33
    },
    "filter": {
        "stopwords": {
            "type": "stop",
            "stopwords": [ " " ]
        }
    },
    "analyzer": {
        "my_ngram_analyzer": {
            "type": "custom",
            "tokenizer": "my_ngram_tokenizer",
            "filter": [ "trim", "stopwords" ]
        }
    },
    "properties": {
        "medicine_id": {
            "type": "integer"
        },
        "medicine_name": {
            "type": "text",
            "analyzer": "standard",
            "search_analyzer": "standard",
            "fields": {
                "ngram": {
                    "type": "text",
                    "analyzer": "my_ngram_analyzer",
                    "search_analyzer": "my_ngram_analyzer"
                }
            }
        }
    }
}

```

logstash 재시작

```
cd /home/ubuntu/dockercompose
docker-compose stop logstash
docker-compose up -d logstash
```

8) Jenkins 설정

<https://i11b305.p.ssafy.io/jenkins> 로 접속

플러그인 설치 후 관리자 계정 생성

Jenkins URL <https://i11b305.p.ssafy.io/jenkins> 로 설정

Dashboard > Jenkins 관리 > Plugins > Available plugins > GitLab, NodeJS 설치
> Jenkins 재시작

Dashboard > Jenkins 관리 > System > GitLab 의 설정을 아래와 같이 설정 후 저장

※ Credentials 는 GitLab에서 발급받은 API 토큰을 사용한다.

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

SaveMe & RescU

GitLab host URL ?

The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com

Credentials ?

API Token for accessing GitLab

GitLab API token (GitLab Access Token) ▼

+ Add ▼

고급 ▼

Test Connection

Dashboard > Tools > Gradle installations 의 설정을 아래와 같이 설정 후 저장

Gradle installations

Gradle installations ^

✎ Edited

Add Gradle

≡ Gradle

✕

name ?

gradle

☒ Install automatically ?

≡ Install from Gradle.org

✕

Version

Gradle 8.9

▼

Add Installer ▼

Dashboard > Tools > NodeJS의 설정을 아래와 같이 설정 후 저장

포팅 매뉴얼

28

NodeJS

Name

nodejs

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 20.16.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Add Installer ▾

Dashboard > 계정 > Credentials 에서 GitLab 계정 Credentials 추가

Dashboard > 계정 > Credentials 에서 GitLab API 토큰 Credentials 추가

Dashboard > 계정 > Credentials 에서 Jenkins ssh key Fingerprint 추가

Dashboard > 계정 > Credentials 에서 security.properties 추가

```
jwt.app.secretkey=xpjpxdozBjxXNZHBF0Lm2aR6gU6qL9eVzbzmQL42gNO
jwt.web.secretkey=xpjpxdozBjxXNZHBF0Lm2aR6gU6qL9eVzbzmQL42gNO
```

Dashboard > 계정 > Credentials 에서 elastic.properties 추가

```
elasticsearch.username=elastic
elasticsearch.password=mByrQq6esenJRy6ypWFR
elasticsearch.uris= elasticsearch:9200
```

Dashboard > 계정 > Credentials 에서 application-prod.properties 추가

```
spring.application.name=smru

server.port=5000

# Spring JPA
spring.jpa.database=mysql
# WARN
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=none
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

spring.data.elasticsearch.repositories.enabled=true
```

Dashboard > 계정 > Credentials 에서 openAPI.properties 추가

```
kakao.api.key=c0d89da28a96548b257283f10df0c84a
kakao.api.url=https://apis-navi.kakao mobility.com
kakao.api.one.path=/v1/directions
kakao.api.many.path=/v1/destinations/directions
kakao.geoCoder.url = https://dapi.kakao.com/v2/local/geo/coor

emergency.api.key=UCK3jCF0xX7Hhb6syYjdncwR2q6N78CMrvLaGvb/N02
emergency.api.url=https://apis.data.go.kr/B552657/ErmctInfoIn
```

Dashboard > 계정 > Credentials 에서 firebase_service_key.json 추가

```
{
  "type": "service_account",
  "project_id": "saveme-ed54f",
  "private_key_id": "c0194d5b5975ea869d96a234ae3cc6d8b842efdf",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBg",
  "client_email": "firebase-adminsdk-9qxyz@saveme-ed54f.iam.g",
  "client_id": "105512064608134432975",
```

```

    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/
    "client_x509_cert_url": "https://www.googleapis.com/robot/v
    "universe_domain": "googleapis.com"
}

```

Dashboard > 새로운 Item > Title에 SpringBoot-CI-CD 입력 > Pipeline 선택 후 OK

GitLab Connection 에서 설정한 SaveMe & RescU 선택

Pipeline 코드 작성

```

pipeline {
    agent any
    tools {
        gradle 'gradle'
    }
    environment {
        GIT_CREDENTIALS_ID = 'gitlab-token' // Jenkins에서 설정
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-be', credentialsId: 'gitlab-
            }
        }
        stage('Grant Permission') {
            steps {
                dir("./backend/smru") {
                    sh 'chmod +x gradlew'
                }
            }
        }
        stage('Copy Security Properties') {
            steps {
                withCredentials([file(credentialsId: 'securit
                    sh 'cp $SECURITY_PROPERTIES ./backend/smr
            }
        }
    }
}

```

```

}
stage('Copy Application Properties') {
    steps {
        withCredentials([file(credentialsId: 'applica
            sh 'cp $APPLICATION_PROPERTIES ./backend/
        }
    }
}
stage('Copy OpenAPI Properties') {
    steps {
        withCredentials([file(credentialsId: 'openAPI
            sh 'cp $OPENAPI_PROPERTIES ./backend/smru
        }
    }
}
stage('Copy Elastic Properties') {
    steps {
        withCredentials([file(credentialsId: 'elastic
            sh 'cp $ELASTIC_PROPERTIES ./backend/smru
        }
    }
}
stage('Copy Firebase Service Key') {
    steps {
        withCredentials([file(credentialsId: 'firebas
            sh 'cp $FIREBASE_SERVICE_KEY ./backend/sm
        }
    }
}
stage('SpringBoot Build') {
    steps {
        dir("./backend/smru") {
            sh './gradlew clean build'
        }
    }
}
stage('deploy') {
    steps {

```



```

        sshagent(credentials: ['jenkins-ssh-key']) {
            sh '''
                ssh -o StrictHostKeyChecking=no ubuntu@i11b305.p.ssafy.io 'cd /var/jenkins_home/workspace/Spring-Boot-2.0.0-RC1'
                scp /var/jenkins_home/workspace/Spring-Boot-2.0.0-RC1/* ubuntu@i11b305.p.ssafy.io:
                ssh -t ubuntu@i11b305.p.ssafy.io 'cd /var/jenkins_home/workspace/Spring-Boot-2.0.0-RC1'
            '''
        }
    }
}
}
}
}

```

Dashboard > 새로운 Item > Title에 Frontend-CI-CD 입력 > Pipeline 선택 후 OK

GitLab Connection 에서 설정한 SaveMe & RescU 선택

Pipeline 코드 작성

```

pipeline {
    agent any
    tools {
        nodejs 'nodejs' // NodeJS 설치 이름을 지정
    }

    environment {
        GIT_CREDENTIALS_ID = 'gitlab-token' // Jenkins에서 설정한 GitLab 토큰 ID
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-fe', credentialsId: 'gitlab-token'
            }
        }

        stage('Check Node') {
            steps {
                sh 'npm -v'
            }
        }
    }
}

```

```

stage('Copy Env') {
    steps {
        withCredentials([file(credentialsId: '.env',
            sh 'cp $ENV ./frontend/mobile/.env'
        )]
        )
        //withCredentials([file(credentialsId: '.env-
        //      sh 'cp $ENV_PROD ./frontend/web/.env'
        //}
        withCredentials([file(credentialsId: '.env-te
            sh 'cp $ENV_TEST ./frontend/web/.env'
        )]
        )
    }
}

stage('Install Dependencies') {
    steps {
        dir('./frontend/mobile') {
            sh 'npm install'
        }
        dir('./frontend/web') {
            sh 'npm install'
        }
    }
}

stage('Build') {
    steps {
        dir('./frontend/mobile') {
            sh 'npm run build'
        }
        dir('./frontend/web') {
            sh 'npm run build'
        }
    }
}

stage('Deploy') {

```

```

        steps {
            sshagent(credentials: ['jenkins-ssh-key']) {
                sh '''
                    ssh -o StrictHostKeyChecking=no ubuntu@i11b305.p.ssafy.io
                    echo "Deploying Mobile..."
                    scp -r ./frontend/mobile/dist/* ubuntu@i11b305.p.ssafy.io:/home/ubuntu/
                    echo "Deploying Web..."
                    scp -r ./frontend/web/dist/* ubuntu@i11b305.p.ssafy.io:/home/ubuntu/
                '''
            }
        }
    }
}
}
}

```

이제 Jenkins를 통해 빌드 및 배포가 진행됩니다.

9) MySQL 덤프파일 불러오기

MySQL WorkBench 실행 > 새 Connection (host: i11b305.p.ssafy.io, port: 3306, 계정정보는 docker-compose.yml 의 내용과 동일)

Server > Data import 를 통해 dump.sql 파일을 불러온 후 실행

10) Jenkins 빌드 및 배포

Jenkins 에서 빌드를 실행하면 서비스가 배포됩니다.