We Fit

빌드 및 배포

2022.07.04 ~ 2022.08.19

대전 공통 2반

유영진 강승훈 김진산 윤영훈 조성환 서준희

1. 기술스택

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 기술스택 | 상세내용 | 버전 |
| 공통 | 형상관리 | GitLab |  |
|  | 이슈관리 | Jira |  |
|  | 커뮤니케이션 | Mattemost |  |
|  | 커뮤니케이션 | Notion |  |
| 서버 | 서버 | AWS EC2 |  |
|  | 플랫폼 | Ubuntu | 20.04 |
|  | 배포 | Docker | 20.10.17 |
|  | 배포 | Docker Compose | 2.6.0 |
|  | 배포 | Jenkins | 2.246.2 |
|  | 배포 | Jenkins | 1.23.1 |
| BackEnd | DB | MySQL | 8..29 |
|  |  | JPA | 5.0.0 |
|  |  | QueryDSL | 5.0.0 |
|  | Java | Zulu | 8 |
|  |  | Spring | 5.3.22 |
|  |  | SpringBoot | 2.7.2 |
|  |  | Spring Security | 2.7.2 |
|  | IDE | Intellij | 2022.2 |
|  | Cache Storage | Redis | 7-alpine |
|  | Build | Gradle | 7.5 |
|  | WebRTC | Openvidu | 2.22.0 |
|  | WebSocket | Stomp | 2.7.2 |
| FrontEnd | HTML5 |  |  |
|  | CSS3 |  |  |
|  | JavaScript(ES6) |  |  |
|  | Vue |  | 3.2.13 |
|  | IDE | Visual Studio Code | 1.70.0 |
|  | WebSocket | Webstomp-client | 1.2.6 |
|  | WebSocket | Sockjs-client | 1.6.1 |

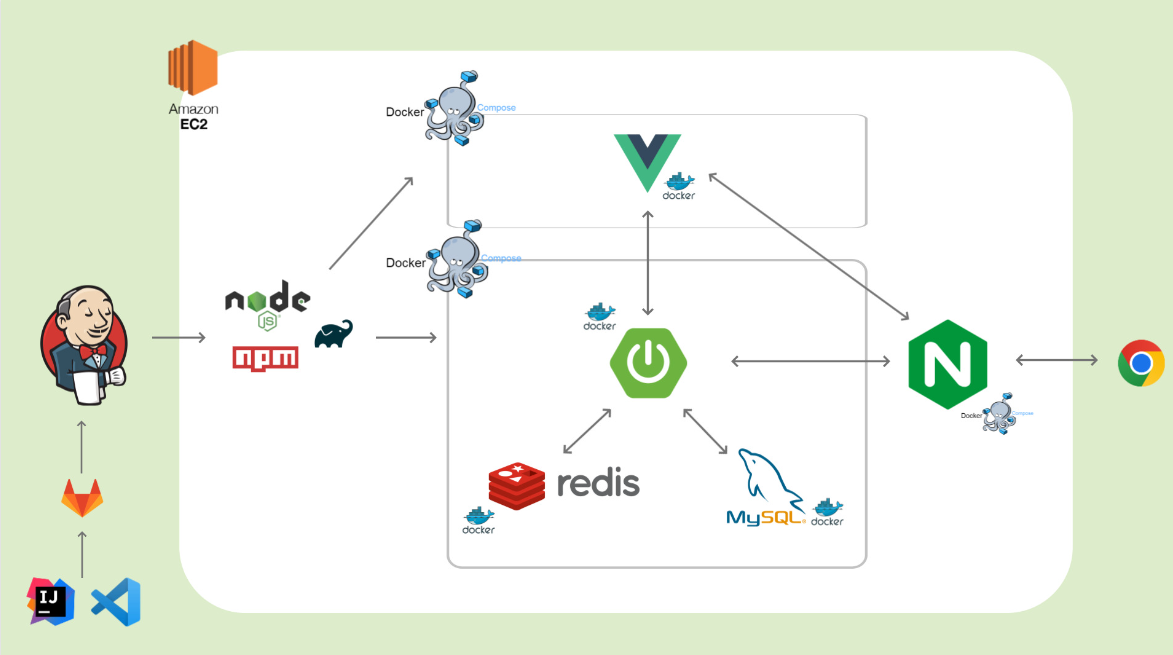
1. 포트번호

|  |  |
| --- | --- |
| 구분 | 포트번호 |
| Vue | 8081 |
| Spring | 8080 |
| Jenkins | 9090 |
| Nginx | 80, 443 |
| Openvidu | 8442, 8443 |
| Redis | 6379 |
| Mysql | 3306 |

1. 상세내용

개요

아래 그림은 서비스의 배포 환경 및 CI/CD 배포 자동화 흐름도이다. 작성한 코드를 GitLab에 Push 또는 Merge하는 경우 WebHook을 이용하여 Jenkins Build Trigger를 통해 Docker Compose가 실행되며 backend, FrontEnd, Nginx가 실행된다.



CI/CD 배포 흐름도

Nginx (confignginx.conf)

Nginx 설정을 위한 파일로 루트 디렉토리 아래에 경로로 작성한다. 모든 도메인으로 요청을 s니 인증서를 이용하여 Reverse Proxy 설정을 한다. 웹 소켓 사용을 위한 헤더 설정 및 에러페이지 처리 등을 설정. Letsencrypt를 사용하여 ssl설정 및 openvidu 사용을 위해 받은 경로를 각각 ssl-certificate, ssl-certificate\_key에 연결한다.

|  |
| --- |
| server {      listen 80;      server\_name i7b206.p.ssafy.io; # 도메인으로 변경      location  {          return 308 https$host$request\_uri;      }  }  server {      listen 443 ssl;      server\_name i7b206.p.ssafy.io;      ssl\_certificate etcletsencryptlivei7b206.p.ssafy.iofullchain.pem;      ssl\_certificate\_key etcletsencryptlivei7b206.p.ssafy.ioprivkey;      location   {          proxy\_intercept\_errors on;          proxy\_pass  http3.36.88.1408081;          add\_header      Content-Security-Policy upgrade-insecure-requests;          error\_page 500 502 503 504 50x.html;          error\_page 404 404.html;      }      location apiv1{          proxy\_intercept\_errors on;          proxy\_hide\_header Access-Control-Allow-Origin;          add\_header 'Access-Control-Allow-Origin' '';  #        add\_header 'Content-Type' 'applicationjson';          proxy\_pass  http3.36.88.1408080$request\_uri;          add\_header  Content-Security-Policy upgrade-insecure-requests;          error\_page 404 404.html;          error\_page 500 502 503 504 50x.html;      }      location ws{          #websocket          proxy\_http\_version 1.1;          proxy\_set\_header Upgrade $http\_upgrade;          proxy\_set\_header Connection upgrade;          proxy\_set\_header Host $host;          proxy\_intercept\_errors on;          proxy\_pass http3.36.88.1408080$request\_uri;          error\_page 404 404.html;          error\_page 500 502 503 504 50x.html;      }      error\_page 404 404.html;      location = 404.html{          root usrsharenginxhtml;          internal;      }      error\_page 500 502 503 504 50x.html;      location = 50x.html{          root usrsharenginxhtml;          internal;      }  } |

Nginx (docker-compose.yml)

Docker에 있는 nginx 이미지를 사용하여 빌드 및 컨테이너를 띄우며 volumes는 빌드된 이미지와 로컬의 경로를 맞춰서 설정파일, ssl 키 등을 읽도록 설정함

|  |
| --- |
| version '3'  services    nginx      image nginxlatest      restart unless-stopped      container\_name nginx      volumes        - .confetcnginxconf.d        - etcletsencryptetcletsencrypt        - .error404.htmlusrsharenginxhtml404.html      ports        - 8080        - 443443  networks    default      external        name nginx-proxy |

Vue (wefitfrontDockerfile)

Alpine에 nodejs를 설치 후 app 디렉토리를 만들고 해당 위치에 package.json을 복사 후 npm을 설치한다. Ignore로 등록된 파일 외에 모두 app으로 복사 후 빌드한다. Nginx를 설치 후 기존 default.conf 파일 제거 후 설정한 경로에 로컬의 default.conf를 복사한다. Html을 저장한 경로의 기존 파일을 삭제하고 build해서 나온 dist 파일을 저장한다.

|  |
| --- |
| FROM nodelts-alpine as builder  # install simple http server for serving static content  RUN npm install -g http-server  # make the 'app' folder the current working directory  WORKDIR app  # copy both 'package.json' and 'package-lock.json' (if available)  COPY package.json .  # install project dependencies  RUN npm install  # copy project files and folders to the current working directory (i.e. 'app' folder)  COPY . .  # build app for production with minification  RUN npm run build  FROM nginxstable-alpine as production  RUN rm -rf etcnginxconf.ddefault.conf  COPY .nginxdefault.conf etcnginxconf.ddefault.conf  RUN rm -rf usrsharenginxhtml  COPY --from=builder appdist usrsharenginxhtml  EXPOSE 8081  CMD [ nginx, -g, daemon off;] |

Vue (wefitfrontendconfdefault.conf)

Nginx로 빌드하여 이미지를 컨테이너에 띄운다.

|  |
| --- |
| server {      listen 8081;      location  {          alias usrsharenginxhtml;          try\_files $uri $uri index.html;      }  } |

Vue (docker-compose.yml)

|  |
| --- |
| version 3.8  services    vue      container\_name vue      build .      restart on-failure      ports        - 80818081      expose        - 8081  networks    default      external        name nginx-proxy |

BackEnd (backendDockerfile)

Java openjdk 8버전을 사용하여 빌드

|  |
| --- |
| FROM openjdk8-jdk-alpine  RUN addgroup -S seungh1024 && adduser -S seungh1024 -G seungh1024  USER seungh1024seungh1024  ARG JAR\_FILE=buildlibs.jar  COPY ${JAR\_FILE} app.jar  ENTRYPOINT [java,-jar,app.jar] |

BackEnd (backendDockerfile)

같은 네트워크 환경을 공유하기 위해 하나의 docker-compose파일에 작성한다.

|  |
| --- |
| version 3.8  services    redis      container\_name redis      image redis7-alpine      ports        - 63796379      volumes        - redisdata    mysqldb      container\_name mysql      image mysqllatest      restart unless-stopped      environment        - MYSQL\_ROOT\_PASSWORD=clsfurtkanth        - MYSQL\_DATABASE=ssafy        - MYSQL\_USER=ssafy        - MYSQL\_PASSWORD=clsfurtkanth      ports        - 33063306      command        - --character-set-server=utf8        - --collation-server=utf8\_unicode\_ci      volumes        - .mysqldatavarlibmysql    spring      container\_name spring      depends\_on        - mysqldb        - redis      build .      restart on-failure      ports        - 80808080      environment        SPRING\_DATASOURCE\_URL jdbcmysqlmysqldb3306ssafyuseSSL=false&useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC&useLegacyDatetimeCode=false        SPRING\_DATASOURCE\_USERNAME root        SPRING\_DATASOURCE\_PASSWORD clsfurtkanth  mysqldb3306ssafyallowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false,  hibernate.dialect.MySQL8InnoDBDialect,      stdin\_open true      tty true  networks    default      external        name nginx-proxy  volumes      redis        driver local |

Jenkins

1. Jenkins 관리 시스템 설정

* Jenkins가 실행중인 서버의 URL 작성
* GitLab URL 작성
* Ssh로 서버에 접속하기 위한 설정
* 프로젝트 빌드 설정 (gradle, nodejs)

1. 프로젝트 생성 & 설정

* Pipeline 생성
* 빌드트리거 설정 (push, merge 시 빌드 유발)
* 소스코드 관리 (pipeline, branch 설정)
* 빌드 환경설정 (pipeline 코드 작성)
* 빌드 후 조치

Openvidu

1. 방화벽 설정

|  |
| --- |
| ufw allow ssh  ufw allow 80/tcp  ufw allow 443/tcp  ufw allow 3478/tcp  ufw allow 3478/udp  ufw allow 40000:57000/tcp  ufw allow 40000:57000/udp  ufw allow 57001:65535/tcp  ufw allow 57001:65535/udp  ufw enable |

1. Openvidu 파일 설치

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

1. 설정

|  |
| --- |
| cd openvidu  nano .env  /\*  DOMAIN\_OR\_PUBLIC\_IP= '주어진 도메인 이름'  OPENVIDU\_SECRET= '비밀키로 사용할 아무 문자열'  CERTIFICATE\_TYPY= letsencrypt  LETSENCRYPT\_EMAIL= '아무 이메일'  HTTP= '포트번호'  HTTPS= '포트번호'  \*/ |

1. 실행

|  |
| --- |
| ./openvidu start |

1. 프로퍼티 정의

MySQL

|  |
| --- |
| mysql.url=jdbc:mysql://mysqldb:3306/ssafy?allowPublicKeyRetrieval-true&useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false mysql.username=ssafy mysql.password=clsfurtkanth |

JWT

|  |
| --- |
| env.jwt.secret=c2lsdmVybmluZS10ZWNoLXNwcmluZy1ib290LWp3dC10dXRvcmlhbC1zZWNyZXQtc2lsdmVybmluZS10ZWNoLXNwcmluZy1ib290LWp3dC10dXRvcmlhbC1zZWNyZXQK |