|  |
| --- |
|  |



|  |
| --- |
| **코딩하는 사람끼리 스터디** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |



목차

[I. 개요 2](#_Toc174632087)

[1. 프로젝트 사용 도구 2](#_Toc174632088)

[2. 개발환경 2](#_Toc174632089)

[- Front-End 2](#_Toc174632090)

[- Back-End 3](#_Toc174632091)

[- Server 3](#_Toc174632092)

[- Database 3](#_Toc174632093)

[3. 외부 서비스 3](#_Toc174632094)

[4. Gitgnore 처리한 핵심 키들 4](#_Toc174632095)

[II. 빌드 4](#_Toc174632096)

[1. 환경변수 4](#_Toc174632097)

[2. 빌드 및 배포 5](#_Toc174632098)

[- Nginx, SSL 5](#_Toc174632099)

[- Git ignore 처리한 핵심 키 파일 7](#_Toc174632100)

[- Jenkins(CI/CD) 9](#_Toc174632101)

[- Openvidu 12](#_Toc174632102)

[- 추가적인 수정 13](#_Toc174632103)

[3. 외부 서비스 적용 14](#_Toc174632104)

[- ChatGpt API 14](#_Toc174632105)

[- STMP (Simple Mail Transfer Protocol) 14](#_Toc174632106)

# 개요

## 프로젝트 사용 도구

* **이슈 관리**
  + JIRA
* **형상 관리**
  + Gitlab
* **커뮤니케이션**
  + Notion
  + Mattermost
* **디자인**
  + Figma
* **UCC**
  + 프리미어 프로
* **CI/CD**
  + Jenkins
* **개발 툴**
  + VS Code 1.90.2
  + IntelliJ IDEA 2024.1.4
* **기타 툴**
  + Postman 11.7.0

## 개발환경

### Front-End

* + Vue.js 3.4.29
    - Vue-router 4.3.3
    - Vuex 4.1.0
  + axios 1.7.2
  + pinia 2.1.7
  + ace-builds 1.35.4
  + animate.css 4.1.1
  + highlight.js 11.10.0
  + Openvidu Browser 2.30.0

### Back-End

* + Java
    - Java OpenJDK 17.0.12 2024-07-16 LTS
    - Spring Boot 3.3.2
      * Spring data JPA 3.3.2
      * Spring Thymeleaf 3.3.2
      * Spring Web 3.3.2
      * Spring Web Services 3.3.2
      * Spring Websocket 3.3.2
      * Spring Webflux 3.3.2
      * Spring Mail 3.3.2
    - MyBatis 3.0.3
    - Jackson 2.16.1
    - Jsoup 1.15.3
    - Lombok 1.18.28
    - Openvidu-java-client 2.30.0
    - MySQL Connector 8.0.32

### Server

* + Ubuntu 22.04
  + Nginx
  + Docker
  + Docker Compose
  + Openvidu 2.30.0

### Database

* + MySQL 8.0.37

## 외부 서비스

* **ChatGPT API**
  + 유료 결재 필요
  + application.properties에서 key 설정
* **STMP (Simple Mail Transfer Protocol)**
  + 보안2단계 인증을 사용하는 Gmail 계정 필요
  + 앱 비밀번호 생성 필요
  + application.properties 에서 설정

## Gitgnore 처리한 핵심 키들

* **Back-End**
  + application.properties
* **Server**
  + docker-compose.yml

# 빌드

## 환경변수

* **Back-End (application.properties)**
  + spring
    - datasource
      * url
      * username
      * password
    - jpa
      * hibernate
        + ddl-auto
    - mail
      * username
      * password
  + openvidu
    - url
    - secret
  + openai
    - api
      * key
* **Front-End (.env)**
  + VITE\_VUE\_API\_URL
  + VITE\_VUE\_SOCKET\_URL

## 빌드 및 배포

### 프로젝트 포트

* + 포트 설정

|  |
| --- |
| sudo ufw allow **8085**  sudo ufw allow **8086**  sudo ufw allow **8087**  sudo ufw allow **8080**  sudo ufw allow **8081**  sudo ufw allow **8082** |

### Nginx, SSL

* + Nginx 설치

|  |
| --- |
| sudo apt-get update  sudo apt-get install nginx –y |

* + Nginx 시작하고 자동으로 시작 되도록 설정

|  |
| --- |
| sudo systemctl start nginx  sudo systemctl enable nginx |

* + SSL 인증서 흭득
    - Certbot 설치

|  |
| --- |
| sudo apt install certbot python3-certbot-nginx -y |

* + - SSL 인증서 발급 및 설정

|  |
| --- |
| sudo certbot—nginx -d <yourdomain> -d <yourdomain> |

* + Nginx 설정 변경
    - bash

|  |
| --- |
| vi /etc/nginx/conf.d/<yourdomain>.conf |

* + - <yourdomain>.conf

|  |
| --- |
| # 이 내용을 <yourdomain>.conf에 붙여넣기  upstream compile\_servers {  server localhost:**8087**; # 컴파일 서버 **1**  server localhost:**8080**; # 컴파일 서버 **2**  server localhost:**8081**; # 컴파일 서버 **3**  }  # HTTP 요청을 HTTPS로 리디렉션  server {  listen **80**;  server\_name i11e108.p.ssafy.io;  location /.well-known/acme-challenge/ {  root /var/www/html;  }  location / {  return **301** https://$host$request\_uri;  }  }  # HTTPS 요청을 포트 **8085**으로 프록시  server {  listen **443** ssl;  server\_name i11e108.p.ssafy.io;  ssl\_certificate /etc/letsencrypt/live/i11e108.p.ssafy.io/fullchain.pem; # managed by Certbot  ssl\_certificate\_key /etc/letsencrypt/live/i11e108.p.ssafy.io/privkey.pem; # managed by Certbot  include /etc/letsencrypt/options-ssl-nginx.conf;  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;  location / {  proxy\_pass http://localhost:**8086**;  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:**8085**; # 백엔드 컨테이너로 요청을 포워딩  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 /socket/ {  proxy\_pass http://localhost:**8085**; # 백엔드 컨테이너로 요청을 포워딩  proxy\_http\_version **1**.**1**;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection "Upgrade";  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 /compiler/ {  proxy\_pass http://compile\_servers; # 로드밸런서  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;  }  } |

* + Nginx 재시작

|  |
| --- |
| sudo systemctl restart nginx |

* + 접속 확인

|  |
| --- |
| https://<yourdomain> |

### Git ignore 처리한 핵심 키 파일

* + docker-compose.yml

|  |
| --- |
| services:  db:  image: mysql:8.0  container\_name: mysql  environment:  MYSQL\_ROOT\_PASSWORD: <your-password>  MYSQL\_DATABASE: <your-database>  volumes:  # - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql  - mysql\_develop\_data:/var/lib/mysql  ports:  - "3306:3306"  networks:  - Cokkiri-develop-Network  backend:  image: ohuggy/cokkiri\_develop\_be  container\_name: Cokkiri-springboot  ports:  - "8085:8080"  depends\_on:  - db  networks:  - Cokkiri-develop-Network  frontend:  image: ohuggy/cokkiri\_develop\_fe  container\_name: vuejs  ports:  - "8086:80"  depends\_on:  - backend  networks:  - Cokkiri-develop-Network  complier\_1:  # image: be-compiler  image: ohuggy/be-compiler  container\_name: compiler\_server\_1  ports:  - "8087:8080"  privileged: true  networks:  - Cokkiri-develop-Network  complier\_2:  # image: be-compiler  image: ohuggy/be-compiler  container\_name: compiler\_server\_2  ports:  - "8080:8080"  privileged: true  networks:  - Cokkiri-develop-Network  complier\_3:  # image: be-compiler  image: ohuggy/be-compiler  container\_name: compiler\_server\_3  ports:  - "8081:8080"  privileged: true  networks:  - Cokkiri-develop-Network  nginx:  image: ohuggy/cokkiri-nginx  container\_name: nginx  ports:  - "8082:80"  depends\_on:  - complier\_1  - complier\_2  - complier\_3  networks:  - Cokkiri-develop-Network  volumes:  mysql\_develop\_data:  networks:  Cokkiri-develop-Network:  driver: bridge |

* + application.properties

|  |
| --- |
| spring.application.name=iscode  # MySQL  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  # DB Source URL  spring.datasource.url=jdbc:mysql:*//db:3306/ssafy*  # spring.datasource.url=jdbc:mysql:*//localhost:3306/ssafy*  # DB username  spring.datasource.username=root  # DB password  spring.datasource.password=<your-password>  spring.jpa.show-sql=**true**  spring.jpa.hibernate.ddl-auto=create  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect  spring.jpa.properties.hibernate.format\_sql=**true**  # mybatis  mybatis.config-location=classpath:config/mybatis-config.xml  # openvidu url  openvidu.url: https:*//<your-domain>:8443/*  # openvidu ***password*** (setting in infra)  # default = MY\_SECRET  openvidu.secret: <your-secret-password>  openai.api.key: <your-openai api key>  spring.mail.host=smtp.gmail.com  spring.mail.port=**587**  spring.mail.username=<your-another-email>  spring.mail.password=tjnp ieqg xajw bhay  spring.mail.properties.mail.smtp.auth=**true**  spring.mail.properties.mail.smtp.timeout=**5000**  spring.mail.properties.mail.smtp.starttls.enable=**true** |

### Jenkins(CI/CD)

* + Jenkins 설치 (ec2 docker와 연동)

|  |
| --- |
| sudo docker run -d --name main\_jenkins --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /home/ubuntu/jenkins-data:/var/jenkins\_home -p **8084**:8080 -u root jenkins/jenkins:jdk17 |

* + - Permission denied docker.sock 에러 해결
      * permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock

|  |
| --- |
| *# 권한 변경*  ls -l /var/run/docker.sock  sudo chown root:docker /var/run/docker.sock  sudo chmod **660** /var/run/docker.sock |

* + Jenkins 접속

|  |
| --- |
| sudo ufw allow **8084**  docker exec -it main\_jenkins /bin/bash  cat /var/lib/jenkins/secrets/initialAdminPassword  *# 여기서 나온 키를 복사 후 http://<your-domain>:8084에 접속 한 후 입력한다.*  *# 이후 추천하는 plug-in을 설치 후 회원가입을 진행* |

* + docker-compose Credentials 생성
    - Jenkins 접속 후 해당 경로로 접속

|  |
| --- |
| Jenkins관리 -> Credentials -> Domains (global)-> add credentials |

* + - 값 설정

|  |
| --- |
| kind: Secretfile  file: docker-compose.yml 찾아서 선택  id: Cokkiri-docker-compose |

* + application-properties Credentials 생성
    - Jenkins 접속 후 해당 경로로 접속

|  |
| --- |
| Jenkins관리 -> Credentials -> Domains (global)-> add credentials |

* + - 값 설정

|  |
| --- |
| kind: Secretfile  file: application-propertise 찾아서 선택  id: Cokkiri-docker-compose |

* + - Item 생성
      * 새로운 item을 pipeline으로 생성
      * 생성된 item에서 pipeline 설정에서 Pipeline script from SCM 선택
      * 값 설정

|  |
| --- |
| SCM: git  Repository URL: 빌드할 git repository url 작성  빌드할 git repository credentials 선택  Branch Specifier: \*/<빌드할 브랜치명>  script Path: Jenkinsfile |

* + - gradle 설정
      * Jenkins 접속 후 해당 경로로 접속

|  |
| --- |
| Jenkins 관리 -> tools -> gradle installations 설정 |

* + - * 값 설정

|  |
| --- |
| name: Cokkiri-gradle  version : Gradle **8.9** |

* + - * gradle 설정이 없는 경우

|  |
| --- |
| Jenkins 관리 -> plug-in -> gradle 설치 |

* + - NodeJs 설정
      * Jenkins 접속 후 해당 경로로 접속

|  |
| --- |
| Jenkins 관리 -> tools -> NodeJS installations 설정 |

* + - * 값 설정

|  |
| --- |
| name: Cokirri-node  version : NodeJS **20.15.0** |

* + - * NodeJs 설정이 없는 경우

|  |
| --- |
| Jenkins 관리 -> plug-in -> NodeJs 설치 |

### Openvidu

* + 설치

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

* + 포트 설정

|  |
| --- |
| sudo ufw allow **22** TCP  sudo ufw allow **80** TCP  sudo ufw allow **443** TCP  sudo ufw allow **3478** TCP+UDP  sudo ufw allow **40000** - **57000** TCP+UDP  sudo ufw allow **57001** - **65535** TCP+UDP  sudo ufw allow **80**  sudo ufw allow **443**  sudo ufw allow **3478**  sudo ufw allow **5442**  sudo ufw allow **5443**  sudo ufw allow **6379**  sudo ufw allow **8888** |

* + 설정 파일 변경
    - bash

|  |
| --- |
| cd openvidu  sudo nano .env |

* + - 해당하는 값 수정

|  |
| --- |
| **DOMAIN\_OR\_PUBLIC\_IP**=<your-domain>  **CERTIFICATE\_TYPE**=letsencrypt  **OPENVIDU\_SECRET**=<your-secret-password>  **HTTP\_PORT**=**8442**  **HTTPS\_PORT**=**8443** |

* + docker-compose.yml 변경
    - bash

|  |
| --- |
| sudo nano docker-compose.yml |

* + - 해당하는 값 수정

|  |
| --- |
| *#in openvidu-server*  - **SERVER\_PORT**=**5443**  *#in nginx*  *#ssl 인증 복사*  - /etc/letsencrypt:/etc/letsencrypt |

### 추가적인 수정

* + Jenkinsfile

|  |
| --- |
| *#1*  stage('Push Docker Image to Docker Hub') {  steps {  script {  docker.withRegistry('https://index.docker.io/v1/', env.DOCKER\_HUB) {  // 백엔드 이미지 푸시  sh 'docker push ${DOCKER\_IMAGE\_BE}'  // 프론트엔드 이미지 푸시  sh 'docker push ${DOCKER\_IMAGE\_FE}'  // 컴파일러 이미지 푸시  sh 'docker push ${DOCKER\_IMAGE\_COM}'  sh 'docker push ${DOCKER\_IMAGE\_NGINX}'  }    }  }  }  *#2*  post {  always {  script {  // Cleanup: Docker Hub에서 로그아웃  sh 'docker logout'  }  }  } |

* + - #1, #2 주석 처리

## 외부 서비스 적용

### ChatGpt API

* + OpenAI 홈페이지에 로그인
  + DashBoard -> API keys로 접속
  + Create new secret key클릭
  + Name을 설정하고 Create secret key 클릭
  + API key 복사 및 저장
  + 사용할 때마다 비용이 청구되므로 주의

### STMP (Simple Mail Transfer Protocol)

* + 메일을 보내는 구글 계정 생성
  + 계정 설정에서 보안 2단계 인증 사용
  + 앱 비밀번호를 기타로 생성
  + application.properties 설정에 값 삽입
    - spring.mail.username : 구글 계정의 이메일
    - spring.mail.password : 앱 비밀번호