



I. 빌드 및 배포

1. 기술스택

형상 관리

- Gitlab

이슈 관리

- Jira 9.6.0

DataBase

- MySQL 8.0.21
- JPA
- QueryDSL

IDE

- Visual Studio Code 1.75.1
- IntelliJ

Front-End

- HTML5
- CSS3
- JavaScript(ES6)
- React 18.2.0
- Styled Components 5.3.6
- axios 1.2.3
- WebSocket
 - stompjs 2.3.3
 - sockjs-client 1.6.1
 - kurento-utils 6.18.0
- Fullcalendar 6.0.3
- MUI 5.11.0

Communication

- Notion 2.20
- KaKao Talk
- Mattermost 7.1.4

OS

- Windows 10

UI / UX

- Figma

Test

- Postman 10.9.4
- JUnit5
- Mockito
- Jacoco toolVersion 0.8.7
- Sonarqube

Back-End

- Java OpenJDK 11.0.16.1
- SpringBoot 2.7.7
- Kurento Media Server 6.18.0
- Kurento
- Gradle 7.5.1

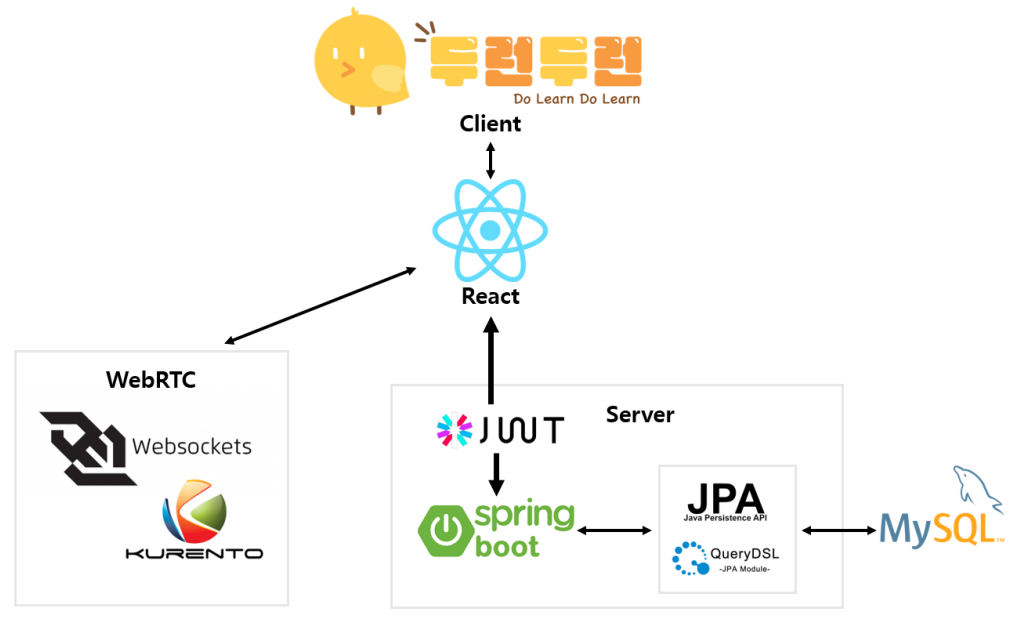
Server

- AWS EC2
- Ubuntu 20.04 LTS
- Docker 20.10.23
- Jenkins 2.375.3
- Nginx 1.18.0
- MobaXterm 23.0

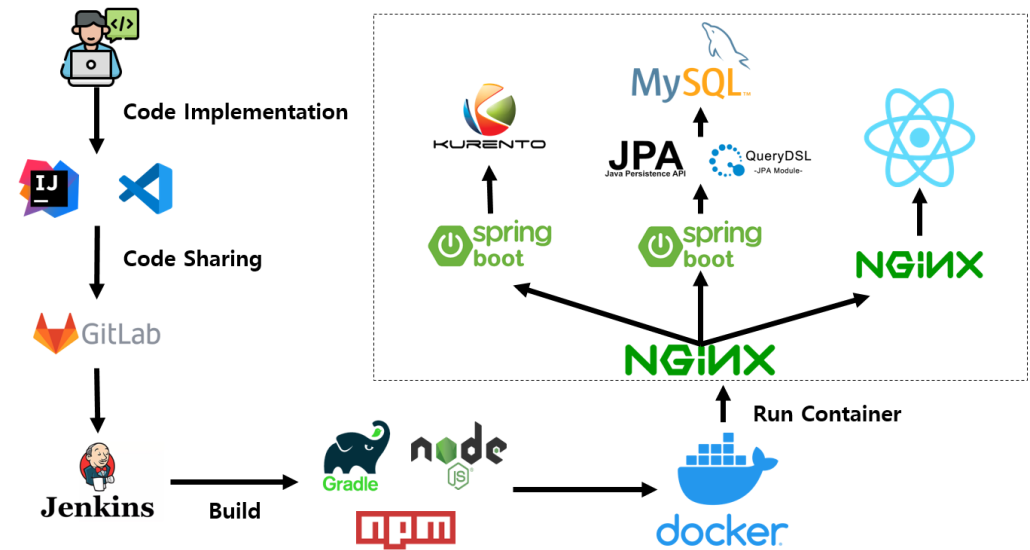
2. 상세 내용

■ 개요

아래 그림은 "두런두런" 서비스의 시스템 아키텍처입니다.



아래 그림은 “두런두런” 서비스의 배포환경 및 CI/CD 배포 자동화 흐름도입니다.



• EC2 초기 세팅

- 타임존 변경하기

```
sudo rm /etc/localtime
sudo ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime
```

- date명령어를 쳤을 때 예시처럼 KST로 변경되면 성공

```
date
ex) Wed Feb 15 10:56:47 KST 2023
```

▪ AWS Docker 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

▪ NGINX 설정

- 서버 패키지 목록 업데이트 & nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
```

- /etc/nginx/nginx.conf 파일 생성후 작성

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
}

http {
    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # SSL Settings
    ##

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: P00DLE
    ssl_prefer_server_ciphers on;

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;

    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
```

```

server {
    listen 80;
    server_name i8a802.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name i8a802.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i8a802.p.ssafy.io/fullchain.pem; # managed by Cert>
    ssl_certificate_key /etc/letsencrypt/live/i8a802.p.ssafy.io/privkey.pem; # managed by Ce>
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Ce
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Cert

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

    location /api { # location 이후 특정 url을 처리하는 방법을 정의
        proxy_pass http://localhost:8080; # Request에 대해 어디

    location /oauth2 {
        proxy_pass http://i8a802.p.ssafy.io:8080;
    }

    location /login/oauth2 {
        proxy_pass http://i8a802.p.ssafy.io:8080;
    }

    location /kurento {
        proxy_pass http://localhost:8888;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Origin "";
    }

    location /oauth-redirect {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

    location /ws {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_send_timeout 240;
        proxy_read_timeout 240;
    }

    location /groupcall {
        proxy_pass https://i8a802.p.ssafy.io:8443;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Origin "";
        proxy_send_timeout 8443;
        proxy_read_timeout 8443;
    }

}
}

```

- **nginx 재시작**

```
sudo service nginx restart
```

- **nginx 상태 확인(running이 뜨면 성공)**

```
sudo service nginx status
```

▪ FrontEnd

- Docker 이미지 생성을 위해 Dockerfile을 작성
(해당 파일은 프로젝트 내에 이미 작성)

```

FROM node:18 as build

WORKDIR /app
COPY package.json .

RUN npm install --save --legacy-peer-deps
COPY . .
RUN npm run build

FROM nginx:alpine

COPY --from=build /app/build /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx/nginx.conf /etc/nginx/conf.d

EXPOSE 3000

CMD ["nginx", "-g", "daemon off;"]

```

- **docker image 생성**

```
docker build -t dolearn-front .
```

- **docker 컨테이너 실행**

```
docker run -d -p 3000:3000 --name front dolearn-front
```

■ BackEnd

■ Backend Dockerfile

```
FROM openjdk:11

ARG JAR_FILE=build/libs/back-end-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java","-Duser.timezone=Asia/Seoul","-jar","/app.jar"]
```

■ docker image 생성

```
docker build -t dolearn-back .
```

■ docker container 생성

```
docker run -d -p 8080:8080 --name back dolearn-back
```

• AWS MYSQL 설정

◦ docker volume 생성

```
docker volume create mysql-volume
```

◦ docker volume 확인

```
docker volume ls
```

◦ 생성된 볼륨으로 마운트하여 임시번호 1234로 컨테이너 실행

```
docker run -d -p 3333:3306 --name mysql -v mysql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD='1234' mysql:8.0.21
```

◦ 컨테이너 내부 진입

```
docker exec -it mysql bin/bash
```

◦ mysql 내부 접속(비밀번호 1234)

```
mysql -u root -p
```

◦ mysql database이동 후 아이디 및 비밀번호 세팅

```
use mysql;
```

◦ root 비밀번호 변경

```
UPDATE user SET authentication_string=PASSWORD('YOURNEWPASSWORD') WHERE User='root';
FLUSH PRIVILEGES;
```

• MySQL

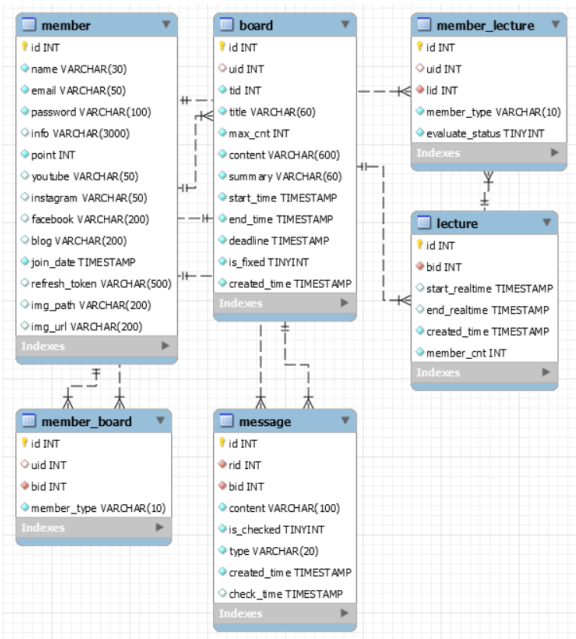



Table	Column	Type	Detail
member	id	INT	Primary Key
	name	VARCHAR	사용자의 이름
	email	VARCHAR	이메일(로그인시 ID로 사용)
	password	VARCHAR	비밀번호
	info	VARCHAR	자기소개
	point	INT	마일리지 포인트
	youtube	VARCHAR	개인 유튜브 링크
	instagram	VARCHAR	인스타 프로필
	facebook	VARCHAR	페이스북 프로필
	blog	VARCHAR	블로그 링크
	join_date	TIMESTAMP	회원가입 일시
	refresh_token	VARCHAR	JWT 로그인시 활용하는 토큰
	img_path	VARCHAR	이미지 서버 경로
Board	img_url	VARCHAR	이미지 요청 경로
	id	INT	Primary Key
	uid	INT	member table을 참조하는 Foreign key
	tid	INT	Thumbnail Image 인덱스
	title	VARCHAR	제목
	max_cnt	INT	수강 정원
	content	VARCHAR	강의 상세 내용
	summary	VARCHAR	강의 내용 요약
	start_time	TIMESTAMP	강의 시작 시간
	end_time	TIMESTAMP	강의 종료 시간
	deadline	TIMESTAMP	모집 기간
	is_fixed	TINYINT	강의 확정 여부
	created_time	TIMESTAMP	요청 게시글 생성 시간

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

- Kind : Username with password
- Scope : Global
- Username : angly97
- Password : dlatlqlqjs!1234 (영타로 임시비번!1234)
- ID : gitlab-access-account

◦ Item 생성

- Enter an item name : dolearn
- Pipeline 으로 생성
- Configure 항목 클릭
 - Git Repository와 연결 설정
 - GitLab Connetion 설정
 - gitlab-connection 선택
 - Use alternative credential 체크
 - Credential을 GitLab API token 선택
 - Build Trigger
 - Build when a change is pushed to GitLab. GitLab webhook URL: http://i8a802.p.ssafy.io:8999/project/dolearn 체크
 - 고급 버튼 클릭
 - 하단의 Secret token 부분에서 Generate 버튼 클릭하여 토큰 생성
 - webhook URL 과 Secret token은 GitLab에서 Webhook 설정 시 필요

• 스크립트 작성

- Pipeline

```
pipeline {
  agent any

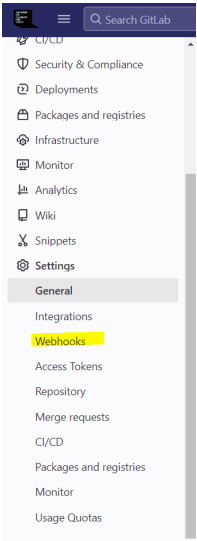
  tools {nodejs "18.12.1-LTS"}

  stages {
    stage('Gitlab') {
      steps {
        git branch: 'main', credentialsId: 'gitlab-access-account', url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12A802.git'
      }
    }
    stage('SpringBootBuild') {
      steps {
        dir('back-end') {
          sh "./gradlew bootJar"
        }
      }
    }
    stage('ReactBuild') {
      steps {
        dir('front-end') {
          sh "npm install --legacy-peer-deps"
        }
      }
    }
    stage('Build') {
      steps {
        sh 'docker build -t dolearn-front ./front-end/'
        sh 'docker build -t dolearn-back ./back-end/'
      }
    }
    stage("Stop and Remove Front Container") {
      steps {
        script {
          def result = sh(returnStdout: true, script: "docker ps -q --filter name=front")
          if (result.trim().length() > 0) {
            sh "docker stop front"
            sh "docker rm front"
            echo "Container named 'front' has been stopped and removed."
          } else {
            echo "No container named 'front' was found."
          }
        }
      }
    }
    stage("Stop and Remove Back Container") {
      steps {
        script {
          def result = sh(returnStdout: true, script: "docker ps -q --filter name=back")
          if (result.trim().length() > 0) {
            sh "docker stop back"
            sh "docker rm back"
            echo "Container named 'back' has been stopped and removed."
          } else {
            echo "No container named 'back' was found."
          }
        }
      }
    }
    stage('Deploy') {
      steps{
        sh 'docker run -d -p 3000:3000 --name front dolearn-front'
        sh 'docker run -d -p 8080:8080 -v /home/ubuntu/dolearn-profile-img:/var/dolearn/images --name back dolearn-back'
      }
    }
    stage('Finish') {
      steps{
        sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
      }
    }
  }
}
```

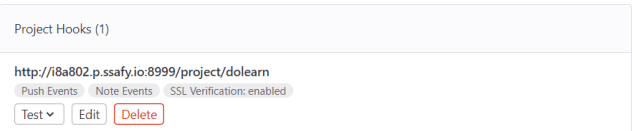
- Deploy 스테이지를 위한 사전 설정
 - back 컨테이너 볼륨 설정을 위한 디렉토리 생성
 - 서버의 /home/ubuntu 경로에, dolearn-profile-img 디렉토리 생성

```
mkdir /home/ubuntu/dolearn-profile-img
```

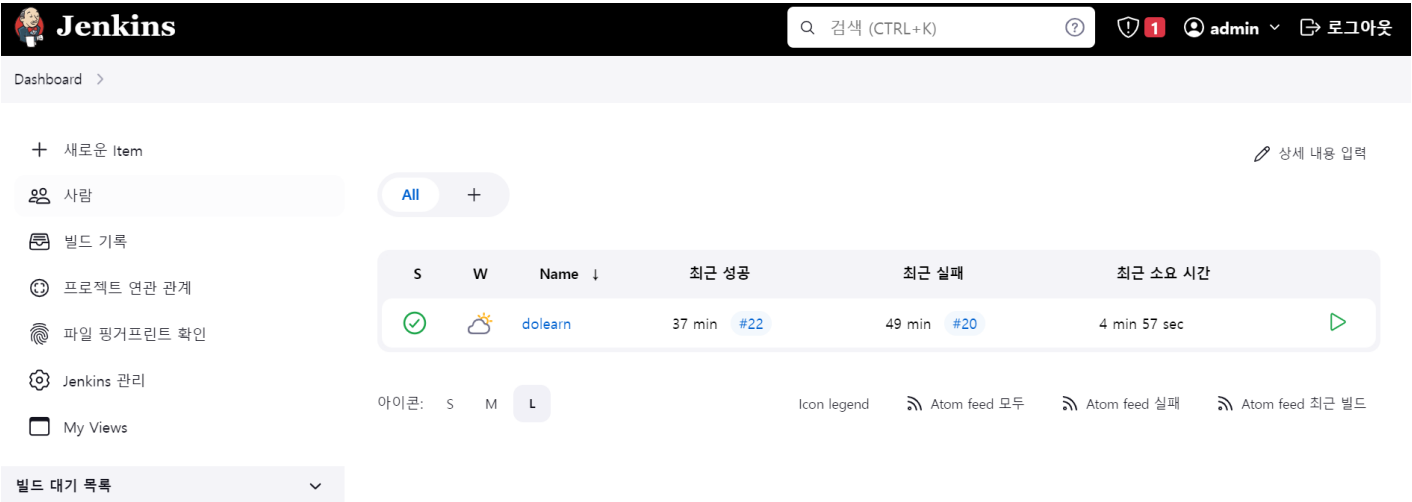
- Webhook 설정
 - gitlab 접속하여 로그인
 - id : angly97
 - password : dlatlqlqjs!1234(임시비번!123)
 - S08P12A802 프로젝트 클릭
 - Settings → Webhook 클릭



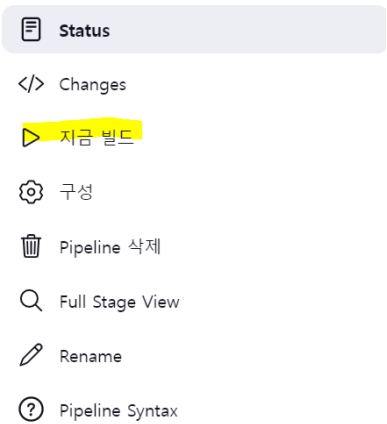
- URL : http://i8a802.p.ssafy.io:8999/project/dolearn
- Secret token : Configure에서 generate한 토큰
- Trigger
 - Push events : main
- 누르고 하단에 생성된 Webhook 확인



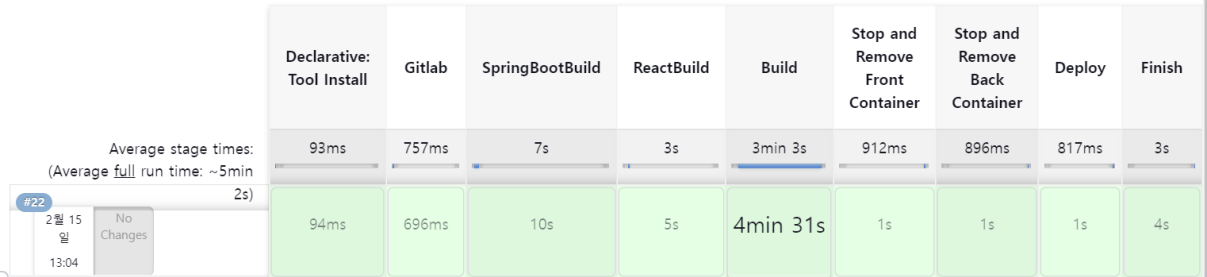
- Jenkins에서 빌드테스트
 - dolearn 클릭



- 지금 빌드 클릭하여 빌드 성공여부 확인



Stage View



- **coturn 설정**

- **coturn 설치**

```
sudo apt-get update && sudo apt-get install --no-install-recommends --yes \ coturn
```

- **/etc/default/coturn 파일 작성**

```
TURNSEVER_ENABLED=1
```

- **현재 서버 public ipv4 알아내기**

```
curl ifconfig.me
```

- **/etc/turnserver.conf 파일 수정**

```
listening-port=3478
tls-listening-port=5349
listening-ip=[ec2 private IP]
external-ip=[ec2 public ip]/[ec2 private ip]
relay-ip=[ec2 private ip]
fingerprint
lt-cred-mech
user=test:test123
realm=myrealm
log-file=/var/log/turn.log
simeple-log
```

- **coturn 가동**

```
sudo service coturn restart
```

- **Kurento-media-server**

- Docker container 실행 (min, max port 지정 후, EC2 Server IP 지정해주기

```
docker run -d -p 8888:8888/tcp -p 49950-50200:49950-50200/udp -e KMS_MIN_PORT=49950 -e KMS_MAX_PORT=50200 -e KMS_STUN_IP=52.79.213.254 -e KMS_STUN_PORT=3478 -e KMS_TURN_URL=test:test123@52.79.213.254:
```

- **Kurento**

- Kurento-client Dockerfile

```
FROM openjdk:11

ENV APP_HOME=/usr/app/

WORKDIR $APP_HOME

COPY build/libs/kurento-0.0.1-SNAPSHOT.jar kurentoApplication.jar

EXPOSE 8443

CMD ["java", "-jar", "-Dkms.url=ws://52.79.213.254:8888/kurento", "kurentoApplication.jar"]
```

- docker image 만들기

```
docker build -t kurento client .
```

- docker container 실행

```
docker run -d -p 8443:8443 --name kurento client kurento-client
```