



Drawing Dream

빌드 및 배포

프로젝트 기간 : 2022.1.10.~2.18

삼성SW청년아카데미 서울캠퍼스 6기

인주비 박기범 손창현 이다예 장준범 제진명

1. 기술스택

구분	기술스택	상세내용	버전
공통	형상관리	Gitlab	-
	이슈관리	Jira	-
	커뮤니케이션	Mattermost, Notion	-
BackEnd	DB	MySQL	5.7
		JPA	-
		QueryDSL	-
	Java	Zulu	8.33.0.1
	Spring	Spring	5.3.6
		Spring Boot	2.4.5
	IDE	Eclipse	JEE 2020-06
	클라우드 스토리지	AWS S3	-
	Build	Gradle	7.3.2
	WebRTC	Kurento Media Server	6.16
		Kurento	-
	API Docs	Swagger2	3.0.0
FrontEnd	HTML5		-
	CSS3		-
	JavaScript(ES6)		-
	React	React	17.0.2
		Redux	7.2.6
		Redux-thunk	2.4.1
	styled-components		5.3.3
	framer-motion		6.0.0
	apexcharts		3.33.0
	toast-ui/react-editor		3.1.2
	toast-ui/react-calendar		1.0.6
	WebSocket	@stomp/stompjs	6.1.2
		stompjs	2.3.3
		sockjs-client	1.5.2
	IDE	Visual Studio Code	1.63.2
Server	서버	AWS EC2	-
	플랫폼	Ubuntu	20.04.3 LTS
	배포	Docker	20.10.12
		Jenkins	2.319.2

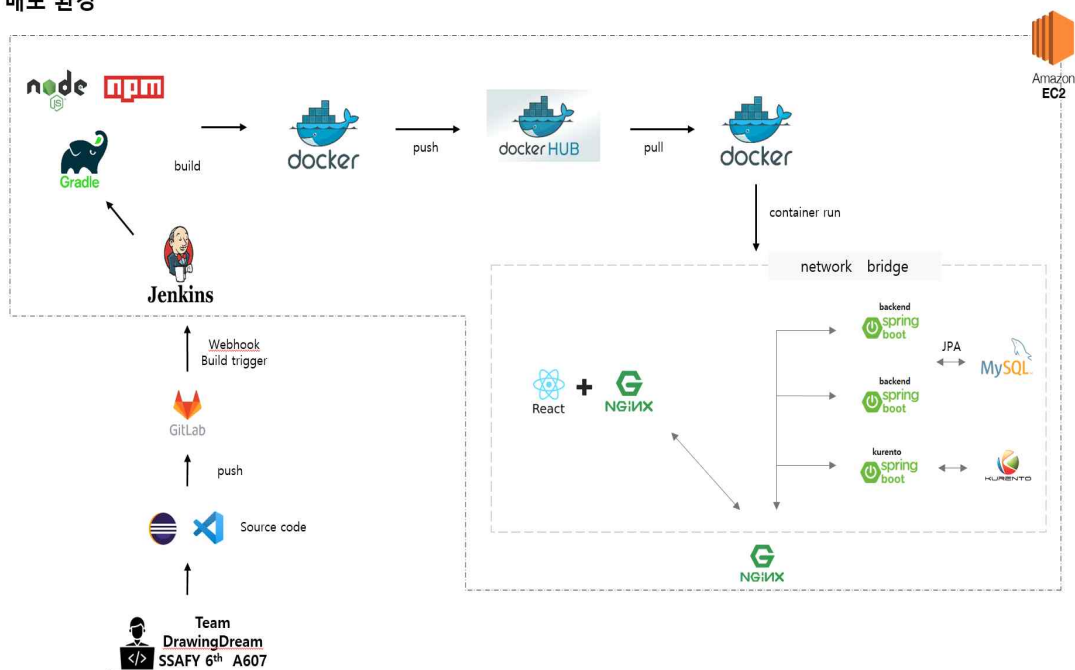
2. 상세내용

□ 개요

아래 그림은 “**DrawingDream**” 서비스의 **배포환경 및 CI/CD 배포 자동화 흐름도**입니다. 팀원들이 각자 작성한 프로젝트를 GitLab에 push 하면, Webhook을 이용하여 Jenkins Build Trigger를 통해 FrontEnd와 BackEnd, Kurento APP을 빌드하게 됩니다.

<CI/CD 배포환경>

배포 환경



각 프로젝트를 빌드 후에는 Docker 이미지를 만들고 이를 Docker Hub에 push 합니다. Docker Hub로부터 서비스에 필요한 이미지를 받아와 컨테이너로 띄웁니다.

SSAFY에서 지원받은 EC2 싱글 인스턴스로 인프라를 구축했습니다. 추후 서비스화를 위해 Nginx는 리버스 프록시 서버로 설정하였습니다. 동시에 Nginx 서버로부터 8444, 8445 포트를 BackEnd 서버로 설정하여 Load Balancing이 가능하도록 구축했습니다.

□ Docker

- docker network 생성

```
docker network create drawingdream-network
```

□ FrontEnd

- Docker 이미지 생성을 위해 Dockerfile을 작성합니다.
(해당 파일은 프로젝트 내에 이미 작성되어있습니다.)

```
from node:lts-alpine as builder

WORKDIR /app
COPY package.json .

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

FROM nginx:alpine

COPY --from=builder /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 이미지 생성

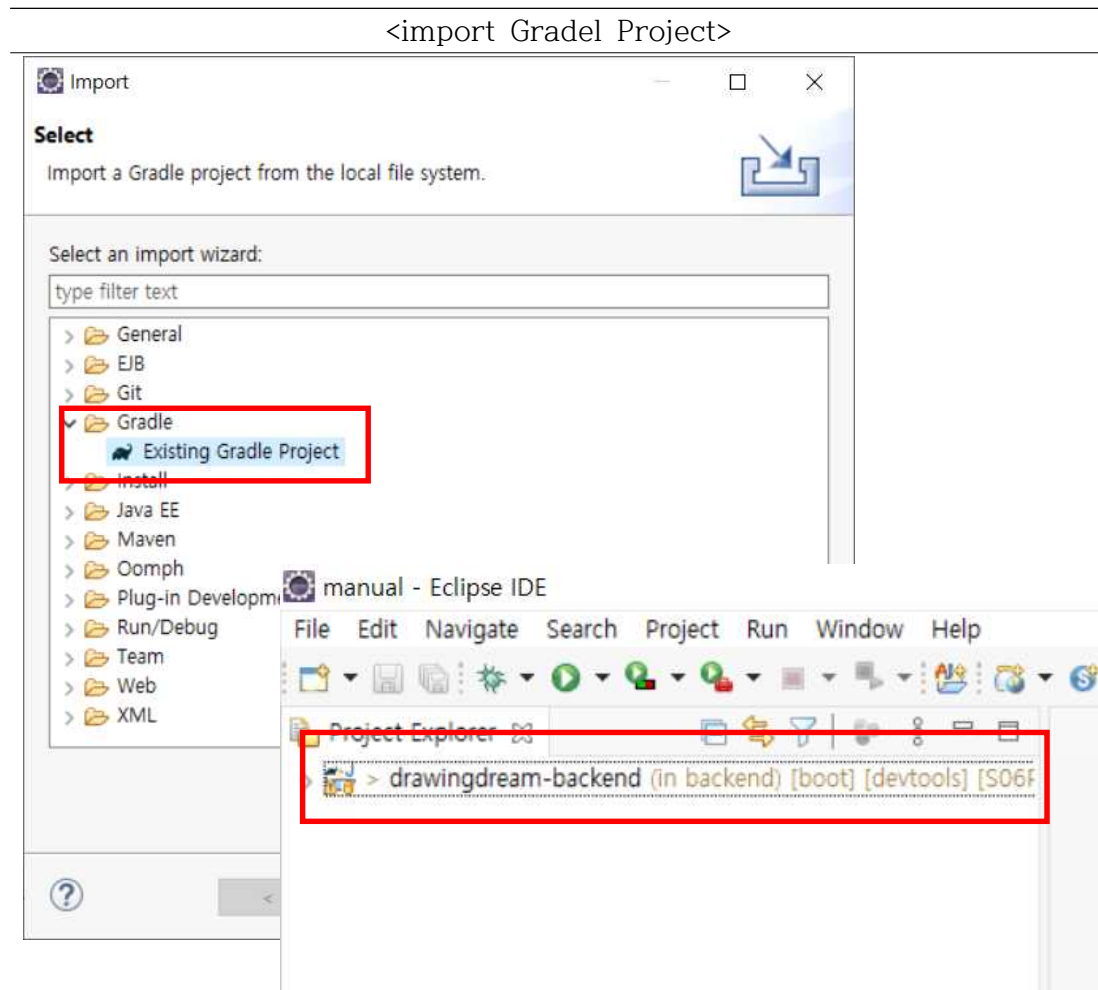
```
docker build -t gganzii1215/drawingdream-frontend:lts -f Dockerfile .
```

- 이미지 컨테이너 실행

```
sudo docker run --network drawingdream-network --name drawingdream-front  
-p 3000:3000 -e TZ=Asia/Seoul -d gganzii1215/drawingdream-frontend:lts
```

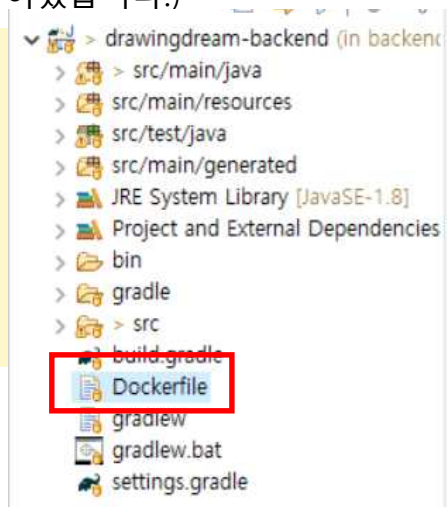
□ BackEnd

- Eclipse에서 backend 폴더를 Gradle로 import 합니다.



- Docker 이미지 생성을 위해 Dockerfile을 작성합니다.
(해당 파일은 프로젝트 내에 이미 작성되어있습니다.)

```
FROM openjdk:8-jre-alpine
ENV APP_HOME=/usr/app/
WORKDIR $APP_HOME
COPY build/libs/*.jar application.jar
EXPOSE 8444
CMD ["java", "-jar", "application.jar"]
```



- Gradle 빌드

```
gradlew clean build -x test --console plain
```

- docker 이미지 생성

```
docker image build -t gganzii1215/drawingdream-backend:its .
```

- mysql 컨테이너 실행

```
sudo docker run --network drawingdream-network -v drawingdream-db-volume:/var/lib/mysql --name drawingdream-db -e MYSQL_ROOT_PASSWORD=drawingdream607! -e TZ=Asia/Seoul -d mysql:5.7
```

- 이미지 컨테이너 실행

```
sudo docker container run --network drawingdream-network --name drawingdream-back -e TZ=Asia/Seoul -p 8080:8080 -p 8444:8444 -d gganzii1215/drawingdream-backend:its
```

// BE 이미지 2개 생성 시, 실행

```
sudo docker container run --network drawingdream-network --name drawingdream-back2 -e TZ=Asia/Seoul -p 8081:8081 -p 8445:8445 -d gganzii1215/drawingdream-backend2:its
```

- 서버 컨테이너 로그 확인

```
docker logs <containerID>
```

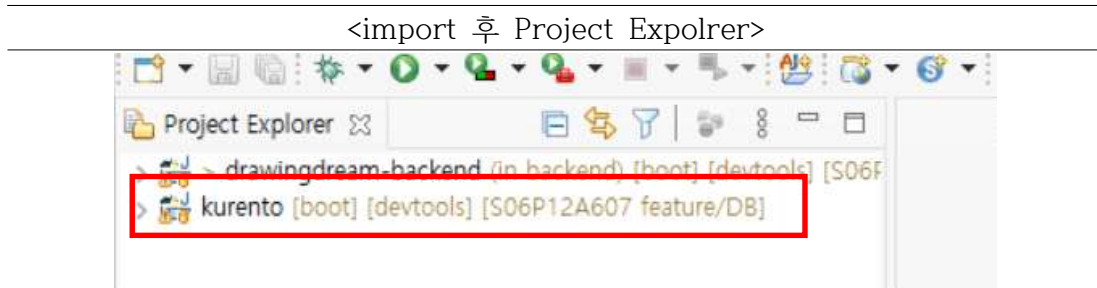
□ Jenkins

- 이미지 컨테이너 실행

```
sudo docker run --network drawingdream-network --name drawingdream-jenkins -p 9090:8080 -v /jenkins:/var/jenkins_home -v /usr/bin/docker:/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -u root -d jenkins/jenkins:its
```

□ Kurento

- Eclipse에서 kurento폴더를 Gradle로 import 합니다.
(Backend 빌드에서의 과정과 동일합니다.)



- Dockerfile 수정

```
COPY build/libs/*.jar kurentoApplication.jar
```

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

- 이미지 컨테이너 실행

```
gradlew clean -x test --console plain
```

```
docker image build -t gganzii1215/drawingdream-kurento:its .
```

```
docker container run --network drawingdream-network --name drawingdream-kurento -p 8443:8443 -e TZ=Asia/Seoul -e JAVA_TOOL_OPTIONS="-Dkms.url=ws://drawingdream-kurento-server:8888/kurento" -d gganzii1215/drawingdream-kurento:its
```

□ Kurento Media Server

- KMS pull 받기

```
docker pull kurento/kurento-media-server:latest
```

- 이미지 컨테이너 실행

```
sudo docker run --name drawingdream-kurento-server --network drawingdream-network -p 8888:8888/tcp -p 5000-5050:5000-5050/udp -e KMS_MIN_PORT=5000 -e KMS_MAX_PORT=5050 -e TZ=Asia/Seoul -e KMS_STUN_IP="3.36.55.8" -e KMS_STURN_PORT="3478" -e KMS_TURN_URL="myuser:mypassword@3.36.55.8:3478?transport=udp" -d kurento/kurento-media-server:latest
```

3. 특이사항

□ 개요

- Drawing Dream 서비스는 Docker 이미지 컨테이너를 기반으로 서비스를 배포하고 있습니다. 서비스에 문제가 발생 시, 아래 명령어를 확인하여 상태를 확인할 수 있습니다. 특히, 온라인수업 서비스에 문제가 발생 시에는 coturn이 정상적으로 구동되고 있는지 확인이 필요합니다. BackEnd, FrontEnd, Kurento, Mysql 프로젝트의 상태를 확인하기 위해선 각 컨테이너의 로그를 확인하는 명령어를 사용하여 log 확인이 가능합니다.

□ Nginx

- 상태 확인

```
sudo service nginx status
```

- 재실행

```
sudo service nginx restart
```

□ Coturn

- 상태 확인

```
sudo service coturn status
```

- 재실행

```
sudo service coturn restart
```


□ Docker

- 컨테이너 확인

```
sudo docker ps -a
```

- 서버 컨테이너 로그 확인

```
docker logs <containerID>
```

- 컨테이너 재실행

```
sudo docker restart <containerID>
```

- 컨테이너 삭제

```
sudo docker rm <containerID>
```

- 이미지 삭제

```
sudo docker rmi <imageID>
```

4. 프로퍼티 정의

□ MySQL

- MySQL Docker 컨테이너에서 DB 스키마를 생성해두면 SpringBoot 구동 시 자동으로 Table 생성 및 Dump Data가 삽입됩니다.
- Spring application.properties DB 관련 설정

```
# Database
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://drawingdream-db/drawingdream_db?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTim
eBehavior=convertToNull&rewriteBatchedStatements=true
spring.datasource.hikari.username=drawingdream
spring.datasource.hikari.password=drawingdream607!
spring.data.web.pageable.one-indexed-parameters=true
spring.datasource.data=classpath:data.sql
spring.datasource.initialization-mode=always
```

- 계정 생성

```
create user 'drawingdream'@'%' identified by 'drawingdream607!';

grant all privileges on *.* to 'drawingdream'@'%' with grant option;
flush privileges;
```

- 스키마 생성

```
create database if not exists drawingdream_db collate utf8mb4_general_ci;
```

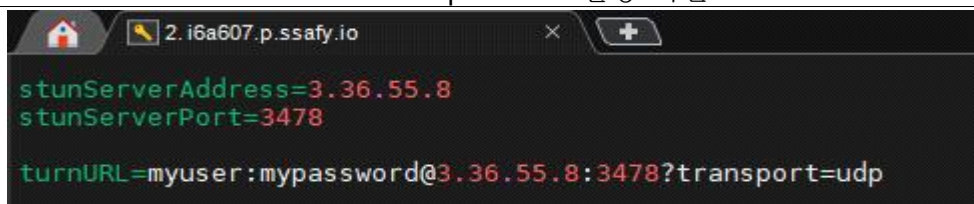
□ Kurento

- WebRtcEndpoint.ini 수정

```
sudo docker exec -it <ContainerID> /bin/bash

vi /etc/kurento/modules/kurento/WebRtcEndPoint.conf.ini
```

<WebRtcEndpoint.ini 설정 화면>



```
stunServerAddress=3.36.55.8
stunServerPort=3478

turnURL=myuser:mypassword@3.36.55.8:3478?transport=udp
```

□ Nginx

○ 환경설정

```
sudo vi /etc/nginx/sites-available/default
```

<Nginx 설정 화면>


```
server {  
    listen 80 default_server;  
    server_name i6a607.p.ssafy.io;  
    if ($host = i6a607.p.ssafy.io) {  
        return 301 https://$host$request_uri;  
    }  
    return 404;  
}  
  
server {  
    listen 443 ssl default_server;  
    server_name i6a607.p.ssafy.io;  
    server_tokens off;  
  
    client_max_body_size 512M;  
  
    client_body_buffer_size 10K;  
    client_header_buffer_size 1k;  
  
    large_client_header_buffers 2 1k;  
  
    ssl_certificate /etc/letsencrypt/live/i6a607.p.ssafy.io/fullchain.pem;  
    ssl_certificate_key /etc/letsencrypt/live/i6a607.p.ssafy.io/privkey.pem;  
    include /etc/letsencrypt/options-ssl-nginx.conf;  
  
    location / {  
        proxy_pass http://i6a607.p.ssafy.io:3000;  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
  
    location /api/ {  
        proxy_pass https://i6a607.p.ssafy.io:8444;  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
  
    location /ws-dd/ {  
        proxy_pass https://i6a607.p.ssafy.io:8444;  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection "upgrade";  
    }  
}
```

□ Coturn

○ 환경설정

```
sudo vi /etc/default/coturn
```

<Coturn 설정 화면>

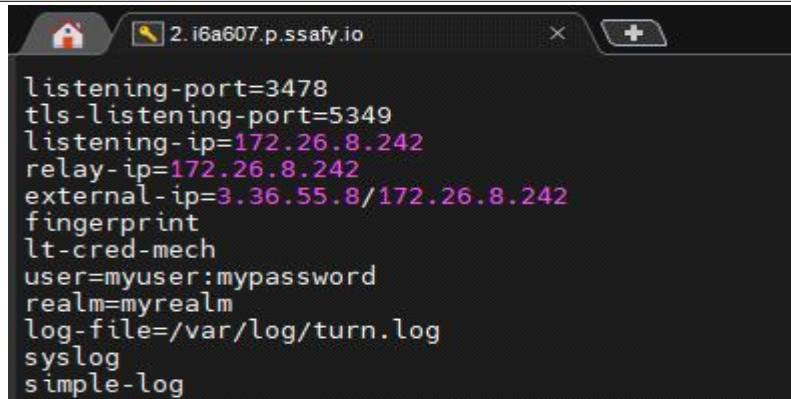
A terminal window with a dark background. The title bar shows a home icon, a search icon, and the text "2. i6a607.p.ssafy.io". The terminal content shows a hash symbol followed by "TURN_SERVER_ENABLED=1".

```
#  
TURN_SERVER_ENABLED=1
```

○ turnserver 환경설정

```
sudo vi /etc/turnserver.conf
```

<turnserver 설정 화면>

A terminal window with a dark background. The title bar shows a home icon, a search icon, and the text "2. i6a607.p.ssafy.io". The terminal content shows various configuration parameters for turnserver.

```
listening-port=3478  
tls-listening-port=5349  
listening-ip=172.26.8.242  
relay-ip=172.26.8.242  
external-ip=3.36.55.8/172.26.8.242  
fingerprint  
lt-cred-mech  
user=myuser:mypassword  
realm=myrealm  
log-file=/var/log/turn.log  
syslog  
simple-log
```