

# GitLab 소스 클론 이후 빌드 및 배포 문서

## 1. 프로젝트 개요

- **프론트엔드:** React + Vite + Nginx
- **백엔드:** Spring Boot (Java 17)
- **AI 서버:** FastAPI (Python 3.12)
- **웹소켓 서버:** Spring Boot 기반 (도커 허브 이미지 사용)
- **배포 방식:** GitLab CI/CD + Jenkins + Docker Compose + Nginx 리버스 프록시
- **클라우드 환경:** AWS EC2
- **데이터베이스:** MySQL (RDS)
- **기타 서비스:** Redis, AWS S3, LiveKit

## 2. 사용된 기술 스택 및 버전

기술	버전
OS	Ubuntu (AWS EC2)
프론트엔드	Node.js 18 + Vite
웹서버	Nginx
백엔드	Spring Boot + OpenJDK 17
AI 서버	FastAPI + Python 3.12
데이터베이스	MySQL (RDS), AWS DynamoDB(NoSQL)
캐시	Redis
CI/CD	GitLab CI/CD + Jenkins
컨테이너 관리	Docker + Docker Compose
HTTPS	Let's Encrypt (Certbot)
웹소켓 서버	Docker Hub 이미지 사용 (Spring Boot 기반)

## 3. 포트 매핑 현황

서비스	컨테이너명	포트 매핑
프론트엔드 (Nginx)	nginx	443 , 7443 , 15080 , 16080 , 28080
백엔드 (Spring Boot)	backend-1 , backend-2	8090 , 8091
FastAPI (AI 서버)	fastapi	8096 (내부 18000)
웹소켓 서버	socket-backend	38080
OpenVidu Server	openvidu-server	기본 OpenVidu 포트 (사용자 설정 필요)
OpenVidu Dashboard	openvidu-dashboard	-
OpenVidu Call	openvidu-call	-
OpenVidu Caddy	openvidu-caddy	-
LiveKit 서비스	ingress , egress	-
모니터링 서비스	grafana , prometheus , loki , promtail	-
Jenkins	jenkins	9191 , 50000

## 4. 환경 변수 관리

### 1) 프론트엔드

- .env 파일은 **GitLab 레포지토리에 포함**

```
VITE_API_BASE_URL=https://www.grimtalk.com/api
VITE_API_AI_COMPARE_IMAGES_URL=https://www.grimtalk.com/fastapi/analysis/compare_images
VITE_LIVE_JOIN_STATUS_URL=https://www.grimtalk.com:28080/live
```

### 2) 백엔드

- .env 파일은 **AWS EC2 ( /home/ubuntu/ )에서 관리**
- .env 예시:

```
DB_URL=
DB_USERNAME=
DB_PASSWORD=
```

```
DYNAMODB_REGION=  
DYNAMODB_TABLE=
```

```
SECRET_KEY=
```

```
AWS_ACCESS_KEY=  
AWS_SECRET_KEY=  
AWS_REGION=  
AWS_BUCKET_NAME=
```

```
REDIS_HOST=  
REDIS_PORT=  
REDIS_PASSWORD=
```

백엔드 환경 변수는 Jenkins 배포 시 EC2 서버에서 직접 로드됨.

## 5. 데이터베이스 (MySQL & DynamoDB)

### 1) MySQL (AWS RDS)

- DB 인스턴스 정보
  - 엔진: MySQL 8.0
  - 호스트:
  - 사용자:
  - 스키마: `grim_talk`
- 테이블 생성은 dump파일을 import 하면 자동으로 됩니다. 그리고 jpa이기에 자동으로 만들어집니다.

### 2) DynamoDB

- 테이블 정보
  - 리전:
  - 테이블명:
  - 파티션 키: `curriculumId` (Long)

- 정렬 키: `time` (int)
- 테이블에 포함된 기타 요소
  - `id` (String)
  - `type` (String)
  - `x` (int)
  - `y` (int)
  - `width` (int)
  - `height` (int)
  - `strokeColor` (String)
  - `backgroundColor` (String)
  - `strokeWidth` (int)
  - `roughness` (int)
  - `opacity` (int)
  - `points` (List<List<Integer>>)
  - `angle` (int)
  - `fillStyle` (String)
  - `strokeStyle` (String)
  - `groupIds` (List<String>)
  - `frameId` (String)
  - `roundness` (Integer)
  - `seed` (Long)
  - `version` (int)
  - `versionNonce` (int)
  - `isDeleted` (boolean)
  - `boundElements` (List<Integer>)
  - `updated` (Long)
  - `link` (String)

- `locked` (boolean)
- `pressures` (List<Integer>)
- `simulatePressure` (boolean)
- `lastCommittedPoint` (List<Integer>)

## 6. OpenVidu 설정

openvidu.env (사용자 정의만 본인이 원하는 값으로 설정, 나머지는 수정 X)

`DOMAIN_NAME`=(사용자 정의)  
`LETSENCRYPT_EMAIL`=(사용자 정의)

`LIVEKIT_API_KEY`=(사용자 정의)  
`LIVEKIT_API_SECRET`=(사용자 정의)

`REDIS_PASSWORD`=(사용자 정의)

`MINIO_ACCESS_KEY`=(사용자 정의)  
`MINIO_SECRET_KEY`=(사용자 정의)

`MONGO_ADMIN_USERNAME`=(사용자 정의)  
`MONGO_ADMIN_PASSWORD`=(사용자 정의)

`DASHBOARD_ADMIN_USERNAME`=(사용자 정의)  
`DASHBOARD_ADMIN_PASSWORD`=(사용자 정의)

`GRAFANA_ADMIN_USERNAME`=(사용자 정의)  
`GRAFANA_ADMIN_PASSWORD`=(사용자 정의)

`EXTERNAL_S3_ENDPOINT`=  
`EXTERNAL_S3_ACCESS_KEY`=  
`EXTERNAL_S3_SECRET_KEY`=  
`EXTERNAL_S3_REGION`=  
`EXTERNAL_S3_PATH_STYLE_ACCESS`=  
`EXTERNAL_S3_BUCKET_APP_DATA`=

ENABLED\_MODULES=observability,app

MONGO\_REPLICA\_SET\_KEY=(사용자 정의)

CADDY\_HTTPS\_PUBLIC\_PORT=(사용자 정의)

CADDY\_HTTP\_PUBLIC\_PORT=(사용자 정의)

CADDY\_RTMP\_PUBLIC\_PORT=1935

CADDY\_MINIO\_PUBLIC\_PORT=9000

CADDY\_HTTP\_INTERNAL\_PORT=7880

LIVEKIT\_TURN\_PUBLIC\_UDP\_PORT=443

LIVEKIT\_TURN\_TLS\_INTERNAL\_PORT=5349

LIVEKIT\_WEBRTC\_PUBLIC\_TCP\_PORT=7881

LIVEKIT\_TURN\_RELAY\_INTERNAL\_PORT\_RANGE\_START=40000

LIVEKIT\_TURN\_RELAY\_INTERNAL\_PORT\_RANGE\_END=50000

LIVEKIT\_WEBRTC\_PUBLIC\_UDP\_PORT\_RANGE\_START=50000

LIVEKIT\_WEBRTC\_PUBLIC\_UDP\_PORT\_RANGE\_END=60000

LIVEKIT\_API\_INTERNAL\_PORT=7780

LIVEKIT\_RTMP\_INTERNAL\_PORT=1945

LIVEKIT\_WHIP\_INTERNAL\_PORT=8080

LIVEKIT\_PROMETHEUS\_INTERNAL\_PORT=6789

LIVEKIT\_INGRESS\_HTTP\_RELAY\_INTERNAL\_PORT=9091

LIVEKIT\_INGRESS\_HEALTH\_INTERNAL\_PORT=9092

LIVEKIT\_INGRESS\_RTC\_UDP\_PORT=7885

LIVEKIT\_EGRESS\_HEALTH\_INTERNAL\_PORT=9093

REDIS\_INTERNAL\_PORT=7000

MINIO\_API\_INTERNAL\_PORT=9100

MINIO\_CONSOLE\_INTERNAL\_PORT=9101

MONGO\_INTERNAL\_PORT=20000

DASHBOARD\_INTERNAL\_PORT=5000

GRAFANA\_INTERNAL\_PORT=3000

LOKI\_INTERNAL\_HTTP\_PORT=3100

LOKI\_INTERNAL\_GRPC\_PORT=9095

DEFAULT\_APP\_INTERNAL\_PORT=6080

## 7. GitLab CI/CD 빌드 및 배포 과정

| Jenkins와 GitLab의 webhook을 이용해 CI/CD 배포

### 1) 전체 배포 흐름

1. GitLab에 `push` → Jenkins가 자동으로 빌드 & 배포 수행
2. Jenkins가 `docker build` 후 이미지 생성
3. `docker-compose.yml` 을 실행하여 컨테이너 배포
4. `nginx.conf` 를 통해 트래픽을 각 컨테이너로 프록시

### 2) 젠킨스 특이사항

→ ssh를 파이프라인에서 사용하기에 플러그인을 설치해야함

→ gitlab을 쓰기 위해서 플러그인을 설치해야함

→ 파이프라인을 레파지토리의 깃랩에 젠킨스파일로 저장해놓았기때문에 젠킨스 내부에 또 적으면 설정이 꼬일 수 있습니다.

## 8. EC2 내부 파일 설정들

### 1) nginx.conf

```
worker_processes auto; # CPU 코어 수에 맞게 워커 프로세스를 자동으로 설정

events {
    worker_connections 1024; # 한 워커 프로세스당 최대 1024개의 연결을 처리할 수
}

http {
    # WebSocket 연결을 위해 업그레이드 헤더 설정
    map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
    }
```

```

}

include /etc/nginx/mime.types; # MIME 타입 설정

# ✅ 백엔드 서버 (로드 밸런싱)
upstream backend_servers {
    least_conn; # 가장 적은 연결을 가진 서버에 요청을 전달 (로드밸런싱 방식)
    server backend-1:8090;
    server backend-2:8090;
}

# ✅ SSE(Server-Sent Events) 서버 (IP 해시 기반 라우팅)
upstream sse_servers{
    ip_hash; # 클라이언트 IP를 기반으로 동일한 서버에 요청을 보내도록 설정
    server backend-1:8090;
    server backend-2:8090;
}

# ✅ FastAPI 백엔드 추가
upstream fastapi_server {
    server fastapi:18000;
}

# ✅ HTTP → HTTPS 자동 리디렉션
server {
    listen 16080;
    server_name i12d202.p.ssafy.io;
    return 301 https://$host$request_uri;
}

# ✅ 🔥 프론트 + 백엔드 + FastAPI 모두 HTTPS 적용
server {
    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl;
    server_name i12d202.p.ssafy.io www.grimtalk.com;

    # ✅ 요청 최대 크기 설정 (1GB)
    client_max_body_size 1024M;

```



```

# ✅ SSL 인증서 설정 (Let's Encrypt 사용)
ssl_certificate /etc/letsencrypt/live/www.grimtalk.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/www.grimtalk.com/privkey.pem;
include /etc/letsencrypt/options-ssl-nginx.conf;
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

# ✅ 루트 경로 요청을 프론트엔드 정적 파일로 리디렉트
location = /home/ {
    return 301 /;
}

# ✅ 🔥 프론트엔드 정적 파일 제공
location / {
    root /usr/share/nginx/html;
    index index.html;
    try_files $uri /index.html; # SPA를 위한 설정

    # 브라우저 캐시 방지
    add_header Cache-Control "no-cache, no-store, must-revalidate";
    add_header Pragma "no-cache";
    add_header Expires 0;
}

# ✅ 🔥 백엔드 API 프록시 설정
location /api/ {
    proxy_pass http://backend_servers;
    proxy_http_version 1.1;
    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;
    proxy_set_header Connection "Upgrade";
}


# ✅ SSE 요청 프록시 (Server-Sent Events)
location /api/sse/ {
    rewrite ^/api/sse/(.*)$ /$1 break;
    proxy_pass http://sse_servers;
}

```

```

    proxy_http_version 1.1;
    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;
    proxy_set_header Connection "";
    proxy_buffering off;
}

```

#  FastAPI 요청 프록시 설정

```

location /fastapi/ {
    rewrite ^/fastapi/(.*)$ /$1 break;
    proxy_pass http://fastapi_server;
    proxy_http_version 1.1;
    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;
    proxy_cache_bypass $http_upgrade;
}

```

```

error_page 404 =200 /index.html;
}

```

#  OpenVidu WebRTC 서버 설정 (포트 7443) -> caddy사용

```

server {
    listen [::]:7443 ssl ipv6only=on;
    listen 7443 ssl;
    server_name i12d202.p.ssafy.io www.grimtalk.com;

    ssl_certificate /etc/letsencrypt/live/www.grimtalk.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/www.grimtalk.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://172.17.0.1:7880; #  WebSocket 요청을 7880으로 전달
        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;

# ✅ 🔥 CORS 허용 추가
add_header Access-Control-Allow-Origin *;
add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
add_header Access-Control-Allow-Headers "Authorization, Content-Ty

# ✅ 🔥 Preflight 요청 지원 (OPTIONS)
if ($request_method = OPTIONS) {
    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods "GET, POST, OPTIONS"
    add_header Access-Control-Allow-Headers "Authorization, Content-
    return 204;
}
}
}

# ✅ 소켓 서버 프록시 설정 (포트 28080)
server {
    listen [::]:28080 ssl ipv6only=on;
    listen 28080 ssl;
    server_name i12d202.p.ssafy.io www.grimtalk.com;

    ssl_certificate /etc/letsencrypt/live/www.grimtalk.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/www.grimtalk.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://socket-backend:38080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $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;
  }
}
}

```

## 2) docker-compose.yml

```

services:
  backend-1:
    image: backend-server # 사용할 Docker 이미지
    container_name: backend-1 # 컨테이너 이름
    restart: always # 컨테이너가 종료되면 자동으로 재시작
    env_file:
      - backend-server.env # 환경 변수 파일을 로드
    ports:
      - "8090:8090"
    networks:
      - backend_network # 컨테이너를 "backend_network" 네트워크에 연결

  backend-2:
    image: backend-server
    container_name: backend-2
    restart: always
    env_file:
      - backend-server.env
    ports:
      - "8091:8090"
    networks:
      - backend_network

  fastapi:
    image: fastapi-server
    container_name: fastapi

```

restart: always

ports:

- "8096:18000"

networks:

- backend\_network

env\_file:

- backend-server.env

socket-backend:

image: jaemoon99/ssp:socket-v5 # 도커허브에 있는 이미지

container\_name: socket-backend

restart: always

env\_file:

- backend-server.env

ports:

- "38080:38080"

networks:

- backend\_network

# 도커 내부에서 호스트의 IP를 참조할 수 있도록 설정

extra\_hosts:

- "host.docker.internal:host-gateway"

nginx:

image: frontend-app # ← 아래 Jenkins 빌드에서 최종 태그로 맞출 예정

container\_name: nginx

restart: always

# 호스트 → 컨테이너 포트 매핑 (원하는 번호 사용)

ports:

- "7443:7443"

- "15080:15080"

- "16080:16080"

- "28080:28080"

- "443:443"

volumes:

- /home/ubuntu/nginx.conf:/etc/nginx/nginx.conf

- /etc/letsencrypt:/etc/letsencrypt

- /home/ubuntu/frontend.env:/usr/share/nginx/.env

extra\_hosts:

```
- "host.docker.internal:host-gateway"

depends_on:
  - backend-1
  - backend-2
  - fastapi

networks:
  - backend_network

networks:
  backend_network:
    driver: bridge
```

**9. 이 파일 정보들을 보고도 잘 모르겠다면 아래의 이메일로 연락주세요.**

**moda2047@naver.com**