



포팅 메뉴얼

1. 프로젝트 기술 스택
2. EC2 설정 시나리오
 - 1) 사용 포트
 - 2) 방화벽 설정
 - 3) Docker 설치
 - 4) Jenkins 설치 및 설정
 - 5) MySQL 설치
 - (1) Ubuntu에 MySQL 설치
 - (2) MySQL Workbench 사용법
 - 6) OpenVidu 설치
 - 7) Nginx 설치
 - 8) 인증서 발급
 - (1) 도메인 발급
 - (2) 인증서 발급
 - 9) OpenVidu 설정
 - 10) Nginx 설정
 - (1) Openvidu Nginx
 - (2) EC2 Nginx
 - (3) Nginx 실행
 - 11) OpenVidu 화면(결과물)
3. 배포
 - 1) 프로젝트의 Dockerfile 및 Nginx 설정
 - (1) Backend - Dockerfile
 - (2) Frontend - Dockerfile
 - (3) Frontend - nginx.conf
 - 2) Jenkins & GitLab 연동
 - (1) jenkins에서 프로젝트 생성
 - (2) 소스 코드 관리
 - (3) 빌드 유발
 - (4) Webhook 설정
 - (5) Build Steps
 - (6) Execute Shell
 - (7) 저장 후 지금 빌드
4. 외부 API
 - 1) 소셜 로그인
 - (1) Kakao
 - (2) Naver
 - 2) AWS S3 Bucket
 - (1) 버킷명
 - (2) 권한 설정 - 버킷정책
 - 3) 랜덤 닉네임 생성 api
 - (1) 출처
 - (2) API Description
 - (3) Example Response

1. 프로젝트 기술 스택

Frontend

- Vue3
- npm : 8.19.3

Backend

- JVM : java-11-openjdk-11.0.15-1
- Spring Boot : 2.7.8

- Node js : 18.13.0
- element-plus : 2.2.28
- IDE : Visual Studio Code

- Gradle: 7.6
- IDE : IntelliJ

DataBase

- MySQL : 5.7.35.0

협업툴

- GitLab
- Jira
- Notion
- MatterMost

Server 및 외부 서비스

- OpenVidu
- Amazon S3
- Amazon EC2
- Nginx : 1.18.0
- Docker : 23.0.0
- Jenkins : 2.375.3

2. EC2 설정 시나리오

1) 사용 포트

- Frontend : 3000
- Backend : 8080
- OpenVidu : 8443
- Nginx : 80, 443
- Jenkins : 9090
- MySQL : 3306

2) 방화벽 설정

1. 현재 방화벽 설정 확인

```
$ sudo ufw status
```

2. 방화벽 설정

- 22(ssh), 443(https), 80(http), 3306(mysql) 열어두기

```
$ sudo ufw allow 22
$ sudo ufw allow 443
$ sudo ufw allow 80
$ sudo ufw allow 3306
$ sudo ufw enable
```

3) Docker 설치

Ubuntu에 도커 설치

```

$ sudo apt-get update

# 필수 패키지 설치
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# GPG Key 인증
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# docker repository 등록
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# 도커 설치
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io

# 도커 확인
$ sudo service docker status

```

4) Jenkins 설치 및 설정

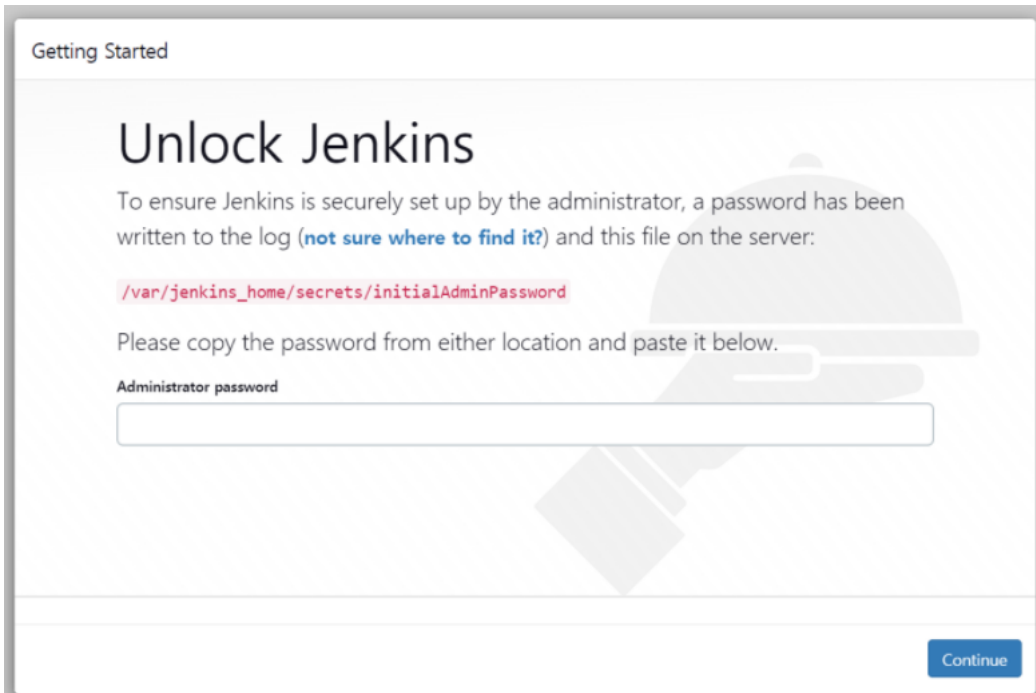
1. jenkins 이미지 설치 및 실행

```
$ sudo docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock
```

이미지 설치 확인

```
$ sudo docker images
```

2. jenkins 접속 (서버 주소)

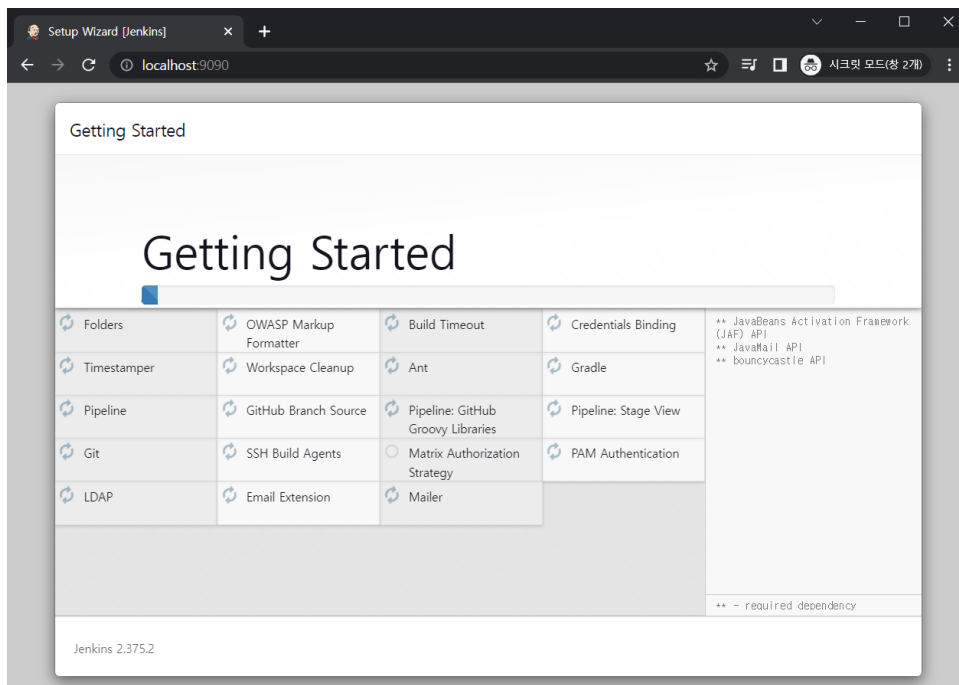
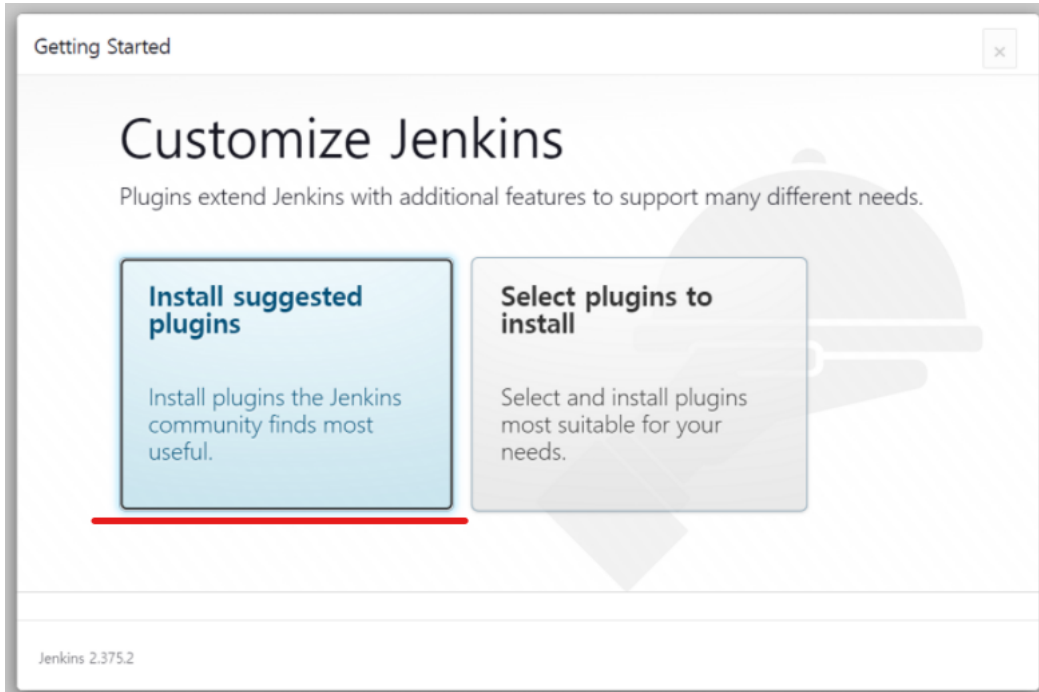


The image shows the Jenkins 'Getting Started' screen. The main heading is 'Unlock Jenkins'. Below it, text explains that a password has been written to the log (not sure where to find it?) and this file on the server: `/var/jenkins_home/secrets/initialAdminPassword`. It asks the user to copy the password from either location and paste it below. There is a text input field labeled 'Administrator password' and a 'Continue' button at the bottom right.

비밀번호 입력하기

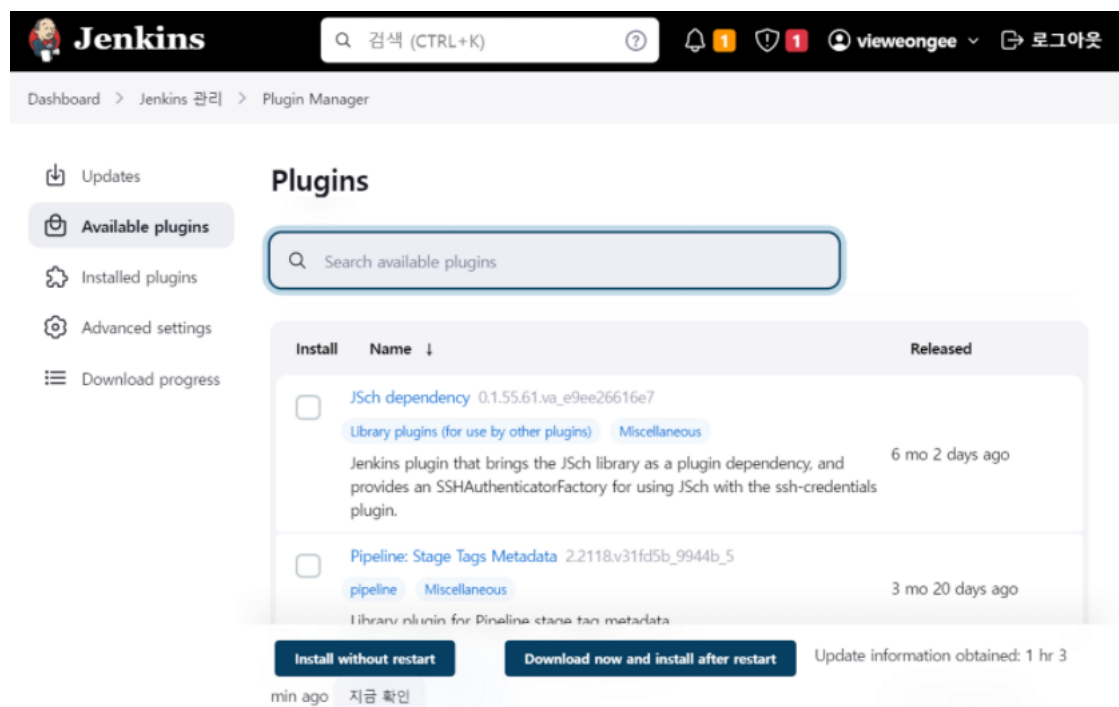
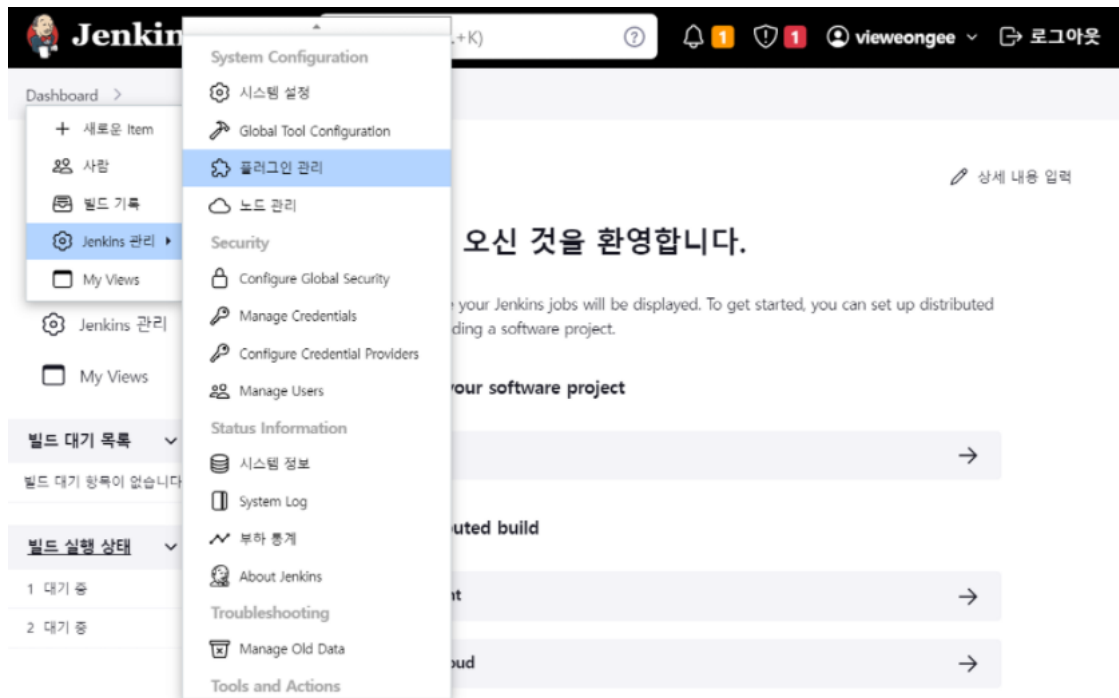
```
# 비밀번호 확인 방법
$ sudo docker logs jenkins
```

3. 기본 설치



4. jenkins 플러그인 설치

Dashboard → Jenkins 관리 → 플러그인 관리 → Available plugins



- Gitlab 관련 항목 설치
 - Gitlab, Generic Webhook Trigger, Gitlab API, Gitlab Authentication 설치
- Docker 관련 항목 설치
 - Docker, Docker Commons, Docker Pipeline, Docker API 설치
- 백엔드에서 Gradle을 사용하였다면 Gradle Plugin도 설치

5. Gradle을 사용하는 경우

- Jenkins 관리 → Global Tool Configuration → 다음과 같이 사용한 Gradle 버전을 맞추고 추가하기

Gradle

Gradle installations

List of Gradle installations on this system

[Add Gradle](#)

Gradle

name ?

vieweongeeGradle

☒ Install automatically ?

Install from Gradle.org

Version

Gradle 7.6

[Add Installer](#)

[Add Gradle](#)

[Save](#) [Apply](#)

6. 젠킨스 컨테이너 안에 도커 설치

```
# 젠킨스 컨테이너 실행
docker exec -it jenkins bash

# 도커 설치 ([EC2] Docker 설치 참조)
```

5) MySQL 설치

(1) Ubuntu에 MySQL 설치

```
# 다음 명령어들을 쳐서 MySQL을 설치
sudo apt-get update
sudo apt-get install mysql-server

# MySQL 구동
sudo systemctl start mysql.service

# MySQL 접속
$ sudo mysql
```

```
# 새로운 유저를 만들기 위한 계정 생성. 이때, GRANT로 필요한 권한을 부여
mysql> CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';

# 어떠한 ip에서든 해당 계정에 모든 권한을 부여한다는 의미.
mysql> GRANT ALL PRIVILEGES ON . TO '계정이름'@'%' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;

# 현재 mysql에서 기본으로 세팅 되어있는 유저들과 추가된 유저를 확인할 수 있음
mysql > SELECT user,authentication_string,plugin,host FROM mysql.user;

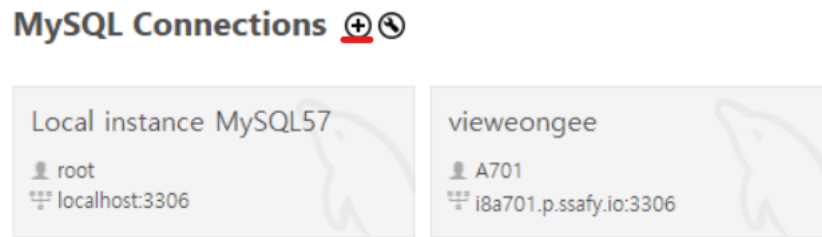
# database 생성(utf8 확장버전. 이모지를 저장할 수 있음.)
mysql > CREATE DATABASE '데이터베이스명' CHARACTER SET utf8mb4 collate utf8mb4_general_ci;

# database의 모든 테이블에 해당 계정이 모든 권한을 행사할 수 있음
mysql > GRANT ALL PRIVILEGES ON '데이터베이스명'.* TO '계정이름'@'%';
```

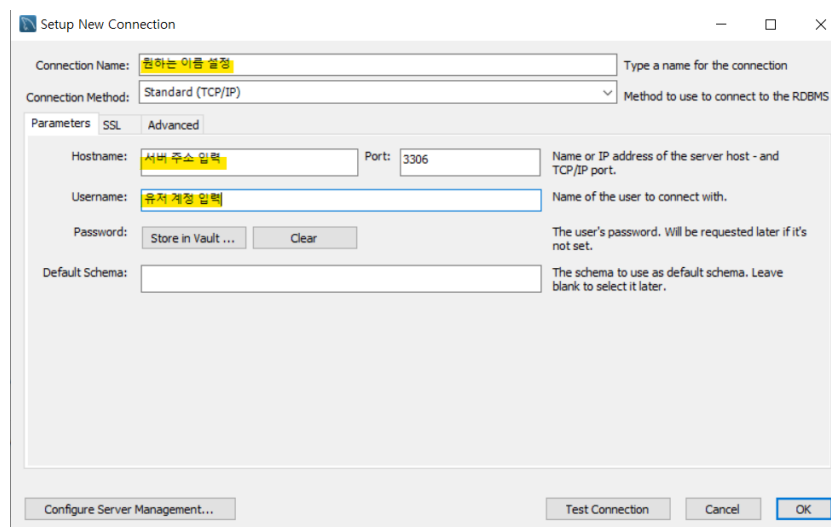
(2) MySQL Workbench 사용법

1. MySQL Workbench 설치
2. Connection 설정

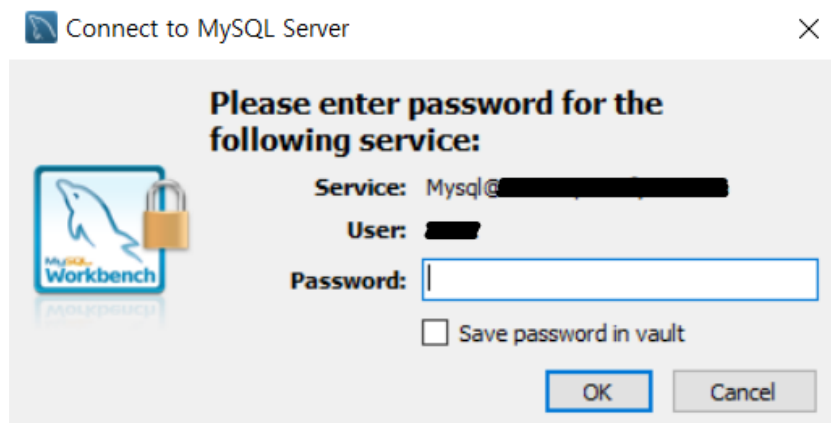
- MySQL Connection에서 '+'를 누르기



- Server에 있는 MySQL과 연결
 - Connection Name: 원하는 이름으로 작성
 - Hostname: 접속할 서버 주소 작성
 - Username: 생성한 MySQL 계정의 username 작성



- 설정한 비밀번호를 입력하면 접속 가능! (비밀번호는 꼭 안전한 것으로 하기)



6) OpenVidu 설치

```
# 루트 계정으로 바꾸기
$ sudo su

# /opt 폴더로 이동
```

```
$ cd /opt

# Openvidu 설치
$ sudo curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | sudo bash

# Openvidu 관련 설정은 SSL 인증서 발급 후에 하기
# Openvidu 포트는 letsencrypt 키를 발급 받기 전까진 기본 포트인 80, 443을 사용해야 하기 때문
# 키를 발급받고 난 후에 포트 변경해도 무방
```

7) Nginx 설치

```
# 설치
sudo apt-get install nginx

# 설치 확인 및 버전 확인
nginx -v

# Nginx 설정은 SSL 인증서 발급 후에 하기
```

8) 인증서 발급

(1) 도메인 발급

ec2 서버의 서버 네임을 `vieweongee.kro.kr` 로 하기 위해 도메인 발급받음

(2) 인증서 발급

```
# letsencrypt 설치
sudo apt update
sudo apt-get install letsencrypt

# 만약 nginx를 사용중이라면 중지
sudo systemctl stop nginx

# 인증서 받기
sudo certbot certonly -d "vieweongee.kro.kr" --manual --preferred-challenges dns
# 이메일 쓰고 Agree
# 뉴스레터 no

# 인증서 위치 폴더 이동
cd /etc/letsencrypt/live/vieweongee.kro.kr

# pem을 PKCS12 형식으로 변경
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name airpageserver -CAfile chain.pem -caname root
```

9) OpenVidu 설정

```
# /opt/openvidu/.env에 설정
$ sudo vi /opt/openvidu/.env
```

```
# 아래 내용 수정
DOMAIN_OR_PUBLIC_IP=vieweongee.kro.kr
OPENVIDU_SECRET=<사용할 비밀번호 입력>
CERTIFICATE_TYPE=letsencrypt
LETSencrypt_EMAIL=vieweongee701@gmail.com
HTTP_PORT=8442
HTTPS_PORT=8443
```


10) Nginx 설정

(1) Openvidu Nginx

기본으로 설정된 그대로 두기 (**절대 건드리지 않기**)

(2) EC2 Nginx

/etc/nginx/conf.d/default.conf와 /etc/nginx/sites-available/default에 설정

```
$ sudo vi /etc/nginx/conf.d/default.conf
$ sudo vi /etc/nginx/sites-available/default
```

```
server {
    location /{
        proxy_connect_timeout    90;
        proxy_send_timeout       90;
        proxy_read_timeout        90;
        proxy_pass http://localhost:3000;
    }

    location /api {
        proxy_connect_timeout    90;
        proxy_send_timeout       90;
        proxy_read_timeout        90;
        proxy_pass http://localhost:8080/api;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/vieweongee.kro.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/vieweongee.kro.kr/privkey.pem;
}

server {
    if ($host = vieweongee.kro.kr) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    server_name vieweongee.kro.kr;
    return 404;
}
```

(3) Nginx 실행

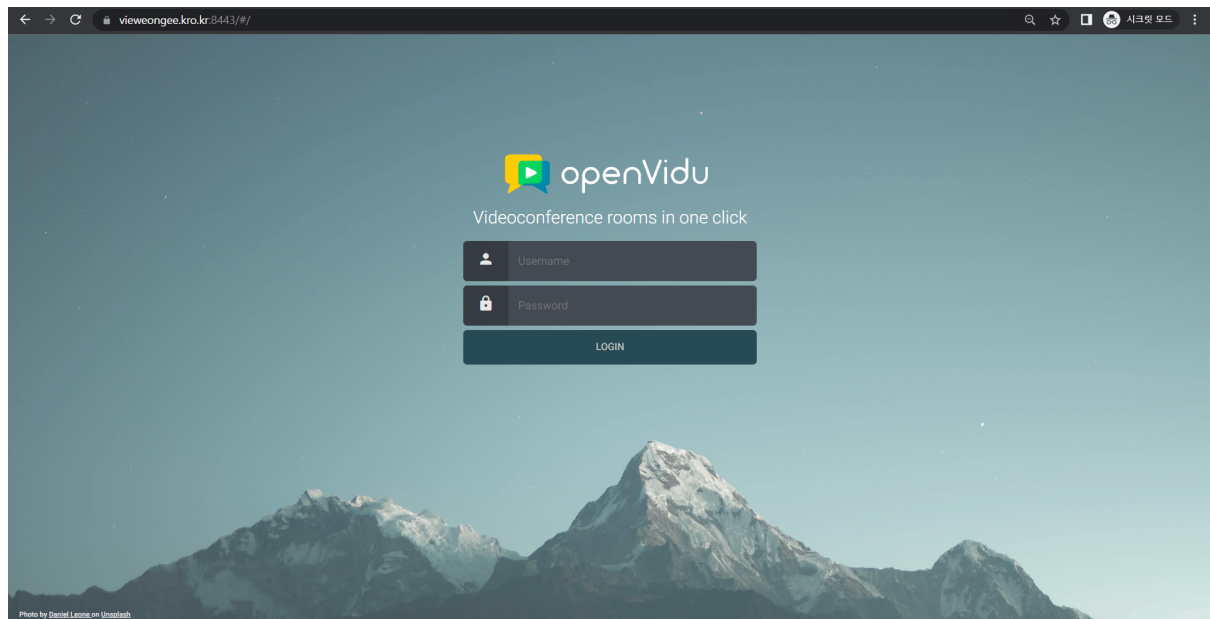
```
# nginx 실행
$ sudo systemctl start nginx

# 실행 확인
sudo systemctl status nginx
```

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-02-13 01:56:25 UTC; 3 days ago
     Docs: man:nginx(8)
  Main PID: 384893 (nginx)
    Tasks: 5 (limit: 19204)
   Memory: 14.7M
   CGroup: /system.slice/nginx.service
           └─384893 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─384894 nginx: worker process
               └─384895 nginx: worker process
                 └─384896 nginx: worker process
                   └─384897 nginx: worker process
```

11) OpenVidu 화면(결과물)

<https://domain:port번호> 로 접속



3. 배포

1) 프로젝트의 Dockerfile 및 Nginx 설정

(1) Backend - Dockerfile

```
# openjdk11로 실행
FROM openjdk:11-jdk

# 해당 경로의 모든 jar파일을 변수로 담기
ARG JAR_FILE=build/libs/*.jar

# 빌드된 jar파일을 api.jar파일 이라는 이름으로 생성
COPY ${JAR_FILE} app.jar

# 컨테이너가 리스닝할 포트
EXPOSE 8080

# 환경 변수 설정
ENV TZ=Asia/Seoul

# 컨테이너를 실행할 때 실행할 커맨드
ENTRYPOINT ["java", "-jar", "app.jar"]
```

(2) Frontend - Dockerfile

```
# node.js로 빌드
FROM node:18 as build-stage

# 경로 설정
WORKDIR /app

# ADD <복사할 파일 경로> <이미지에서 파일이 위치할 경로>
ADD . .

# 의존성 설치
RUN npm install
```

```
# 빌드 -> dist 폴더 생성됨
RUN npm run build

# nginx로 실행
FROM nginx:stable-alpine as production-stage

# 컨테이너가 리스닝할 포트
EXPOSE 3000

# nginx.conf를 default.conf로 복사
COPY ./nginx/nginx.conf /etc/nginx/conf.d/default.conf

# /app/dist를 /usr/share/nginx/html로 복사
COPY --from=build-stage /app/dist /usr/share/nginx/html

# 컨테이너를 실행할 때 실행할 커맨드
CMD ["nginx", "-g", "daemon off;"]
```

(3) Frontend - nginx.conf

```
server {
    # 포트 번호
    listen 3000;

    # 경로 설정
    location / {
        # index 파일이 있는 경로
        root /usr/share/nginx/html;

        # index 파일로 지정할 파일 설정
        index index.html index.htm;

        # 요청한 주소의 uri를 무시하고 index.html 파일을 제공
        try_files $uri $uri/ /index.html;
    }
}
```

2) Jenkins & GitLab 연동

(1) jenkins에서 프로젝트 생성

: Dashboard → 새로운 Item → 프로젝트 이름 입력 → Freestyle project

(2) 소스 코드 관리

소스 코드 관리 > Git을 선택하고 git clone 주소 입력

- Credentials 아래의 Add 버튼을 클릭해서 깃 아이디와 비밀번호 저장
- 저장한 credential을 클릭했을 때 에러메세지가 뜨지 않으면 정상 접근 연동에 성공

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

[Redacted]

Credentials ?

[Redacted]

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

(3) 빌드 유발

- Build when a change is pushed to GitLab 체크

(4) Webhook 설정

1. Jenkins project 선택 > 구성 > 빌드 유발
 - 꼭 저장 눌러주기!

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: [REDACTED] ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)

☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

Generate

Clear

2. GitLab Settings > Webhooks

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☐ Merge request events

A merge request is created, updated, or merged.

SSL verification

☒ Enable SSL verification

Save changes

Test ▾

Delete

Push events

Tag push events

Issues events

Confidential issues events

Note events

Confidential note events

Merge requests events

Job events

Pipeline events

	Elapsed time	Request time	
	0.04 sec	1 hour ago	View details
	0.02 sec	1 hour ago	View details
	0.02 sec	3 hours ago	View details
	0.01 sec	3 hours ago	View details

3. GitLab Webhooks의 URL과 Secret token에 Jenkins의 URL과 Secret token을 작성한다.
4. GitLab에서 main branch에 push할 때마다 자동 빌드 되게 설정해준다.
5. 저장 후, Test버튼을 눌러 원하는 테스트를 해볼 수 있다.

(5) Build Steps

Build Steps

Invoke Gradle script ?

Invoke Gradle ?


Gradle Version

vieweongeeGradle

☐ Use Gradle Wrapper ?

Tasks ?

clean build



고급...

(6) Execute Shell

```
# frontend 컨테이너 생성
docker build -t vieweongee_frontend:latest ./frontend

# 이미 실행 중인 frontend 컨테이너가 있다면 중단하기
if (docker ps | grep vieweongee_frontend) then docker stop vieweongee_frontend; fi

# frontend 컨테이너 실행
docker run -d --rm --name vieweongee_frontend -p 3000:3000 vieweongee_frontend

# backend 컨테이너 생성
docker build -t vieweongee_backend:latest ./backend
```


```
# 이미 실행 중인 backend 컨테이너가 있다면 중단하기
if (docker ps | grep vieweongee_backend) then docker stop vieweongee_backend; fi

# backend 컨테이너 실행
docker run -d --rm --name vieweongee_backend -p 8080:8080 vieweongee_backend

# 중단한 뒤 남아있는 이미지들 삭제
docker image prune -f
```

(7) 저장 후 지금 빌드

빌드 추이로 확인 가능


Build History
추이 ▼

/

✓ #126

2023. 2. 16. 오전 6:33
Started by GitLab push by

✓ #125

2023. 2. 16. 오전 6:20
Started by GitLab push by

✓ #124

2023. 2. 16. 오전 4:23
Started by GitLab push by

✓ #123

2023. 2. 16. 오전 4:22
Started by GitLab push by

✓ #122

2023. 2. 16. 오전 3:48
Started by GitLab push by

✓ #121

2023. 2. 16. 오전 3:39
Started by GitLab push by

4. 외부 API

1) 소셜 로그인

(1) Kakao

- 기본 정보

기본 정보

앱 ID	855738
앱 이름	vieweongee
사업자명	뷰영어

- redircet uri

```
http://localhost:8080/api/login/oauth2/code/kakao
https://vieweongee.kro.kr/redirect
http://vieweongee.kro.kr:8080/api/login/oauth2/code/kakao
```

- 수집 정보

- 닉네임, 프로필사진, 카카오계정 (이메일)

개인정보

항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	설정
프로필 사진	profile_image	● 필수 동의	설정
카카오계정(이메일)	account_email	● 선택 동의 [수집]	설정

(2) Naver

- 기본 정보

뷰영어

개요	API 설정	네이버 로그인 검수상태	멤버관리	로그인 통계	
----	--------	-----------------	------	--------	--

애플리케이션 정보

Client ID	hCg3zNttO1i50cUABPON
Client Secret 보기

- redirect uri

```
http://vieweongee.kro.kr:8080/api/login/oauth2/code/naver
https://vieweongee.kro.kr/redirect
http://localhost:3000/
http://localhost:8080/api/login/oauth2/code/naver
```

- 수집 정보

- 이메일 주소

선택하세요.

▼

네이버 로그인

제공 정보 선택(이용자 식별자는 기본 정보로 제공) [?]

필수 항목은 개인정보보호법 제3조 제1항, 제16조 제1항 등에 따라 서비스 제공을 위해 필요한 최소한의 개인정보만을 선택해야 합니다.

권한	필수	추가
회원이름	<input type="checkbox"/>	<input type="checkbox"/>
이메일 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>
별명	<input type="checkbox"/>	<input type="checkbox"/>
프로필 사진	<input type="checkbox"/>	<input type="checkbox"/>
성별	<input type="checkbox"/>	<input type="checkbox"/>
생일	<input type="checkbox"/>	<input type="checkbox"/>
연령대	<input type="checkbox"/>	<input type="checkbox"/>
출생연도	<input type="checkbox"/>	<input type="checkbox"/>
휴대전화번호	<input type="checkbox"/>	<input type="checkbox"/>

사용 API

×

2) AWS S3 Bucket

(1) 버킷명

- vieweongee

Amazon S3 > 버킷

▶ 계정 스냅샷

Storage Lens는 스토리지 사용량 및 활동 추세를 대한 가시성을 제공합니다. 자세히 알아보기 [?](#)

Storage Lens 대시보드 보기

버킷 (1) [Info](#)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기 [?](#)

↺

ARN 복사

비어 있음

삭제

버킷 만들기

🔍 이름으로 버킷 찾기

< 1 > ⚙️

이름	AWS 리전	액세스	생성 날짜
<input type="radio"/> vieweongee	아시아 태평양(서울) ap-northeast-2	퍼블릭	2023. 2. 5. pm 1:41:23 PM KST

(2) 권한 설정 - 버킷정책

```
{
  "Version": "2012-10-17",
  "Id": "Policy",
  "Statement": [
    {
      "Sid": "Stmt",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::vieweongee/*"
    }
  ]
}
```

3) 랜덤 닉네임 생성 api

(1) 출처

- <https://nickname.hwanmoo.kr/>

(2) API Description

GET <https://nickname.hwanmoo.kr/?format=json&count=2>

Parameter	Required	Description
format	Yes	json, text
count	No	1, 2, 3, ..., 1000
max_length	No	should be greater than 6
whitespace	No	_ , - or whatever you want
seed	No	

(3) Example Response

```
{
  "words": ["결혼한 종이상자", "즐고있는 실리콘밸리"],
  "seed": "hwanmoo.yong"
}
```