

# 빌드 및 배포 가이드

## 프로젝트 기술 스택 - 버전 및 툴

- 백엔드
  - SpringBoot
  - sts-3.9.14.RELEASE
  - Gradle 6.8.3
  - JVM 11.0.16 (Ubuntu 11.0.16+8-post-Ubuntu-0ubuntu120.04)
  - JPA
  - openjdk version "11.0.16"  
OpenJDK Runtime Environment (build 11.0.16+8-post-Ubuntu-0ubuntu120.04)  
OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
  - MySQL 8.0.30 / MySQL Workbench 8.0
- 프론트엔드
  - VSCode
  - React
  - Openvidu 2.22.0 OnPremises
  - Node v10.19.0 / NPM v6.14.4
  - nginx 1.18.0

## SSL 인증서 발급 by Cert-Bot

- Certbot으로 SSL 인증서를 발급받기 위해선 NginX 가 선행으로 설치되어야함
  - 우분투 접속 후
    1. `sudo apt update`  
`sudo apt upgrade` 를 통해 apt 업그레이드 진행
    2. `sudo apt install nginx` 를 통해 NginX 설치

설치 경로는 etc/nginx

sudo service start nginx 를 통해 nginx 실행한 다음

netstat -lntp 통해 80번 포트 리스닝 되는지 확인하고

127.0.0.1 로 접속해서 구동 확인 - Welcome to nginx! 뜨면 성공!

◦ Certbot 으로 SSL 인증서 발급

1. sudo add-apt-repository ppa:certbot/certbot

sudo apt-get install python-certbot-nginx 를 통해 certbot 설치

2. sudo certbot --nginx -d i7a402.p.ssafy.io 통해 도메인에 https 적용

[Ubuntu, Nginx] Let's Encrypt로 https 적용하기

개인적인 용도로 사용하는 서버라면 상관 없지만, 회원가입  
과 같은 개인정보가 사이트에 들어가는 경우 2012년 8월 18  
일부터 법이 변경되어 정보통신망 이용촉진 및 정보보호 등

☹️ <https://syudal.tistory.com/entry/Ubuntu-Nginx-Lets-Encrypt%EB%A1%9C-https-%EC%A0%81%EC%9A%A9%ED%95%98%EA%B8%B0>



◦ 발급받은 인증서를 이용해 PKC12 파일 생성 (Springboot 프로젝트 내 HTTPS 적용 시 필요)

1. sudo su 를 통해 관리자 권한을 얻은 후 SSL 인증서 있는 경로로 이동

⇒ /etc/letsencrypt/live/i7a402.p.ssafy.io/

2. openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name tomcat -CAfile chain.pem -caname root

⇒ pkc12 파일 생성 후 비밀번호 입력 (기억해둘것!)

3. 스프링부트 프로젝트 내 applicationProperties 파일과 같은 경로에 pkc12 키 복사

4. applicationProperties 파일 내 속성 수정 (만약 있다면 알맞은 정보로 수정)

```
server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=아까입력한비밀번호입력
server.http2.enabled=true
```

## OPENVIDU 설치

1. `sudo su` ⇒ 관리자 권한으로 변경한 뒤
2. `cd /opt` ⇒ openvidu가 설치되는 경로로 이동
3. `curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash`  
를 통해 openvidu on promises 설치
4. `exit` 으로 나가서 `cd openvidu` 로 설치 폴더로 이동

⇒ 여기까지 해서 설치는 끝. 이제 SSL을 위한 설정을 변경해주자

5. `cd /opt/openvidu` 로 폴더 이동 후 `sudo vi .env` 로 환경설정 수정
  - `DOMAIN_OR_PUBLIC_IP=<도메인 또는 public IP>` ⇒ 미리 발급받은 도메인 입력
  - `OPENVIDU_SECRET=MY_SECRET` ⇒ 아무거나 입력해줘도 되지만 기억해둘 것
  - `CERTIFICATE_TYPE=letsencrypt` ⇒ 인증서를 certbot으로 발급받았다면 letsencrypt로 변경
  - `LETSENCRYPT_EMAIL=user@example.com` ⇒ 만약 인증서 타입이 letsencrypt라면 이메일 설정
  - `HTTP_PORT=8081 / HTTPS_PORT=8443` ⇒ NGINX의 포트를 변경
6. `./openvidu start` 를 통해 실행

⇒ `https://<설정한 도메인 or IP>:8443/` ⇒ 여기서 접속확인해보자

\*\*\* 만약 인증서가 제대로 동작하지 않는다면? `sudo rm -rf certificates` \*\*\*

#### [OpenVidu]EC2 + React + Openvidu 적용기


Openvidu를 이용해 webRTC를 구현하기 위해선 일단 websocket 연결을 위해 Frontend에 https가 적용되어야 한다. ubuntu 20.04 LTS 기존 openvidu 관련 docker image가 존재한다면 모두 삭제해  
🐾 <https://skdltm117.tistory.com/90>

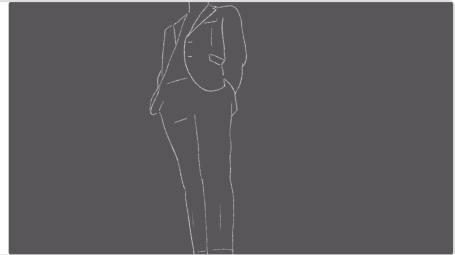


## DOCKER 설치

- <https://download.docker.com/linux/ubuntu/dists/>  
접속 후 버전에 맞는 설치파일 받는다 (.deb)
- `sudo dpkg -i /path/to/package.deb` 를 통해 파일 설치

### [Docker] Ubuntu에 Docker 설치하기

커널 버전 확인: 리눅스 커널이 최소 3.10 버전 이상이어야 한다.  
아래와 같이 확인 가능하다. \$ `uname -r` >> " 3.10.0-  
327.13.1.el7.x86\_64 " sudo 권한 혹은 root 권한을 소유한 계  
 <https://dongle94.github.io/docker/docker-ubuntu-install/>



## MySQL 설치 및 설정

1. `docker pull mysql:latest` ⇒ MySQL 도커 이미지 파일 다운로드
2. `sudo docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=비밀번호 --name my_mysql_server mysql` 로 도커에 MySQL 실행
3. `sudo docker exec -i -t my_mysql_server bash` ⇒ 도커에 실행중인 MySQL 접속
4. `mysql -u root -p` 엔터 후 위에서 입력한 비밀번호 입력해서 MySQL root 계정 접속
  - a. `use mysql;` ⇒ `select user, host from user;` ⇒ 유저 정보 확인
  - b. `CREATE USER 'ssafy'@'%' IDENTIFIED BY '비밀번호';` ⇒ 유저 만든다
  - c. `GRANT ALL PRIVILEGES ON . TO 'ssafy'@'%';` ⇒ 앞에서 만든 회원에게 권한 부여
5. 이후 로컬 환경에서 워크벤치 실행 후 아래 사진처럼 설정해준다
  - a. Connection Name : 임의 설정
  - b. Connection Method : Standard(TCP/IP) ⇒ 기본값
  - c. Hostname : i7a402.p.ssafy.io ⇒ 가상서버 호스트네임
  - d. Port : 3306 ⇒ MySQL 기본 포트
  - e. Username : ssafy ⇒ 아까 만든 유저 이름
  - f. Password : 비밀번호 ⇒ 아까 만든 비밀번호

입력 후 TestConnection 해보자

\*\*\*\* 만약 안된다면 도커에 실행중인 MySQL 에 접속해서 restart 해보자 \*\*\*\*

Setup New Connection

Connection Name:  Type a name for the connection

Connection Method:  Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname:  Port:  Name or IP address of the server host - and TCP/IP port.

Username:  Name of the user to connect with.

Password:   The user's password. Will be requested later if it's not set.

Default Schema:  The schema to use as default schema. Leave blank to select it later.

## Git Clone 후 배포

```
git clone https://lab.ssafy.com/s07-webmobile1-sub2/S07P12A402.git
```

⇒ 깃으로 옮기면 수정할 예정

각 프로젝트를 도커화 시켰으므로 각각의 도커 파일만 실행시켜주면 될...것...

(⇒ 도커파일에 키 복사하는 코드 경로 제대로 해주자)

- Front
  - 도커없이 그냥 실행하려면
    - a. `sudo apt-get install nginx` ⇒ NginX 설치
    - b. `cp /home/ubuntu/FrontEnd/nginx.conf /etc/nginx/site-available/nginx.conf`
      - ⇒ 첨부된 nginx.conf 설정 파일을 설치한 NginX 설정으로 복사
    - c. `sudo service nginx start` ⇒ NginX를 이용해 React 프로젝트 실행

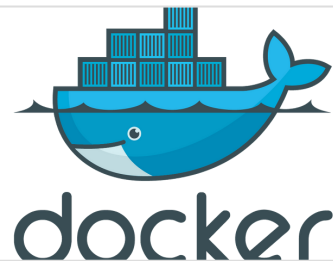
- Front 폴더에서 npm init ⇒ npm install 차례로 실행
- docker build -t nginx-react:0.1 . ⇒ nginx-react 라는 이름의 도커 이미지 생성하고 (이름 바뀌어도 노상관)  
⇒ 아마 여기까지 끝내고 완성된 도커파일만 올릴듯
- docker run --name nginx\_react -d -p 3000:80 nginx-react:0.1  
⇒ 도커로 배포하면 3000번 포트로 접속 가능  
\*\* 접속 정보는 nginx.conf 파일 / 배포 정보는 Dockerfile 파일에 있으므로 참고할 것 \*\*
- Back
  - **도커없이 그냥 실행**하려면 백엔드 jar 파일이 있는 경로에서 (통상 프로젝트 내 build/libs)  
⇒ nohup java -jar 압축파일이름 &     백그라운드에서 백엔드 서버가 실행된다
  - docker build -t blink\_backend:0.1 . ⇒ blink\_backend 라는 이름의 도커 이미지 생성하고(이름 바뀌어도 노상관)
  - docker run --name blink -d -p 8081:8081 blink\_backend:0.1  
⇒ 도커로 배포하자
- Node - server.js + Dockerfile 만 가지고도 돌릴수있게 해보자
  - 그냥 node server.js 해버리면 되긴 하는데... 도커로 되는 버전을 작성해보자
  - **도커없이 그냥실행**하려면 server.js 가 있는 경로로 이동해서
    - cp /etc/letsencrypt/live/i7a402.p.ssafy.io/cert.pem  
/home/ubuntu/FrontEnd/src/testserver/cert.pem 실행 후  
cp /etc/letsencrypt/live/i7a402.p.ssafy.io/privkey.pem  
/home/ubuntu/FrontEnd/src/testserver/privkey.pem 실행  
⇒ cert-bot 을 통해 발급받은 SSL 인증서를 node서버 사용을 위해 복사
    - nohup node server.js & ⇒ 백그라운드에서 노드 서버가 실행된다
  - server.js 가 있는 경로에서 npm init ⇒ 패키지이름만 docker-node 로 설정 후 엔터 연타

- package.json 파일 생긴거 확인한 다음
- npm install --save express    npm install --save socket.io    npm install --save cors  
⇒ 필요한 패키지들 설치 한 다음
- sudo docker build -t docker-node .    ⇒ 도커로 빌드
- sudo docker images    ⇒ 방금 빌드한 이미지 확인 후
- sudo docker run -p 4000:4000 docker-node    ⇒ 실행

#### [Docker] Docker로 서버 구축(node.js)

실제 서비스 중인 서버를 운용하고 있다면 그 서버를 실수로라도 날리는 위험은 절대로 있어서는 안될 것입니다. 서버를 구축하는데 있어서 '안정성'은 굉장히 중요한 요소입니다. 도

 <https://ebbnflow.tistory.com/206>



## 명령어

- ps -ef | grep '이름일부' : 백그라운드에서 실행중인 프로젝트 확인  
⇒ 보통 nohup 을 통해 백그라운드에서 실행되고 있는 백엔드랑 노드 서버를 찾는데 쓰임  
⇒ kill -9 찾은아이디    로 종료시킨다
- iptables -L : Ubuntu 의 네트워크 환경 조회