

# 서울\_4반\_A402\_설치형\_포팅매뉴얼

## Instruction

- EC2 혹은 Lightsail 등으로 이용 가능한 Ubuntu 서버가 필요합니다.
  - 원활한 설치를 위해 서버 기본 상태에서 시작하는 것을 권장합니다.
- TOOLIV Community Edition 서버 사양은 다음과 같습니다.  
최소한 아래 명시 사양 이상의 서버를 사용해서 설치할 것을 권장합니다.
  - Memory 16GB
  - 4 vCPUs (4 코어)
  - Storage(EBS Volume) 10GB 이상

## Prerequisites

원활한 설치와 설정을 위해 root 권한으로 진행합니다.

```
1 sudo su
```

ubuntu 에서 설치 가능한 패키지 리스트를 업데이트합니다.

```
1 apt update
```

TOOLIV 배포에 필요한 여러 패키지를 설치합니다.

- **vim 에디터 설치**
  - 파일을 수정하기 위해 vim을 설치합니다.

```
1 apt install -y vim
```

- **git** 설치
  - TOOLIV Github를 클론 받기 위해 설치합니다.

```
1 apt install git
```

- **nodejs** 설치
  - client 프로젝트 빌드를 위해 nodejs 및 npm을 설치합니다.

```
1 curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -  
2 apt-get install -y nodejs
```

- **NGINX** 설치
  - 리버스 프록시 NGINX 서버를 설치합니다.

```
1 apt install -y nginx
```

NGINX 설정 파일을 letsencrypt - certbot 에서 자동 설정할 수 있도록 구동시킵니다.

```
1 service nginx start
```

TOOLIV은 **React client**, **Spring Boot Server**, **OpenVidu**, **redis** 및 **NGINX** 를 사용하므로  
아래 항목들에 대한 인바운드 규칙 을 설정해줍니다.

TCP	UDP
22	
80	
443	
3000	
3478	3478
8443	
40000 - 57000	40000 - 57000
57001 - 65535	57001 - 65535

## Install Certbot

 본 가이드는 **Ubuntu 20.04.4** 기준으로 작성된 예시입니다.

Certbot 설치 관련하여 자세한 내용을 확인하고 싶다면,  
[certbot.eff.org](https://certbot.eff.org) 혹은 [Certbot 참고문서](#) 확인해주세요.

Certbot 설치를 위해 snap을 이용하여 core를 설치합니다. core는 snap 최신 버전을 유지하기 위해 설치합니다.

```
1 snap install core
2 snap refresh core
```

만약 기존에 Certbot이 설치되어 있다면 Certbot을 삭제합니다.

```
1 apt remove certbot
```

snap을 이용하여 Certbot을 설치합니다.

```
1 snap install --classic certbot
```

Certbot 명령을 로컬에서 실행할 수 있도록 연결시켜줍니다.

```
1 ln -s /snap/bin/certbot /usr/bin/certbot
```

아래 명령어를 실행하여 Certbot을 이용해 SSL 인증서를 받아온 후, Certbot이 NGINX를 설정합니다.

```
1 certbot --nginx
```

서버 환경에 맞도록 NGINX를 설정합니다.

```
1 vim ${path}/etc/nginx/sites-available/default
```

아래 내용은 참고할 만한 예시입니다. ( etc/nginx/sites-available/default - 443 server 블록 )

포트 번호는 그대로 이용하는 것을 권장합니다.

```
1 location / {
2     proxy_pass http://도메인:클라이언트 컨테이너 포트(ex: 3000);
3     proxy_set_header Host $http_host;
4     proxy_set_header X-Real-IP $remote_addr;
5     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
6 }
7
8 location /api/ {
9     proxy_pass https://도메인:서버 컨테이너 포트(ex: 8443);
10    proxy_set_header Host $http_host;
11    proxy_set_header X-Real-IP $remote_addr;
12    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
13 }
14
15 location /chatting/ {
16    proxy_pass https://도메인:서버 컨테이너 포트(ex: 8443);
17    proxy_set_header Host $http_host;
18    proxy_set_header X-Real-IP $remote_addr;
19    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
20
21    proxy_http_version 1.1;
22    proxy_set_header Upgrade $http_upgrade;
23    proxy_set_header Connection "upgrade";
24 }
25
26 location /openvidu/ {
27    proxy_pass https://도메인:오픈비두 HTTPS 포트(ex: 4443);
28    proxy_set_header Host $http_host;
29    proxy_set_header X-Real-IP $remote_addr;
30    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
31 }
32
```

발급받은 SSL을 서버에 적용하기 위해 tooliv.p12 파일을 export 합니다.

```
1 openssl pkcs12 -export -in ${path}/etc/letsencrypt/live/{도메인}/cert.pem -inkey
```

추가로 비밀번호 입력하는 창이 나면 tooliv.p12 키에 대한 비밀번호를 입력합니다.

❗ 비밀번호는 **Build And Run Project** 과정에서 필요한 정보로, 메모해두시기 바랍니다.

## Clone Github repository

Github Tooliv repogitory에서 오픈소스 코드를 clone 받아옵니다. ( home/ubuntu 경로에서 실행 )

```
1 git clone https://github.com/TOOLIV/Tooliv.git
```

**server** 프로젝트 폴더 내에 있는 **tooliv.p12** 파일을 삭제합니다.

```
1 rm -rf ${path}/home/ubuntu/Tooliv/server/src/main/resources/keystore/tooliv.p12
```

생성한 **tooliv.p12** 파일을 프로젝트 내 폴더로 복사합니다.

```
1 cp ${path}/home/ubuntu/tooliv.p12 ${path}/home/ubuntu/Tooliv/server/src/main/resources/
```

## Execute Shell script

TOOLIV 배포에 필요한 패키지를 자동으로 설치하고, 필요한 Docker 볼륨과 네트워크를

편리하게 생성하기 위해 shell script 파일을 **home/ubuntu** 경로로 가져옵니다.

```
1 cp ${path}/home/ubuntu/Tooliv/tooliv-install.sh ${path}/home/ubuntu
```

tooliv-install.sh 파일에 실행 권한을 부여합니다.

```
1 chmod 755 ${path}/home/ubuntu/tooliv-install.sh
```

tooliv-install.sh 파일을 실행하여 패키지 설치 및 환경설정을 자동으로 실행합니다.

```
1 ${path}/home/ubuntu/tooliv-install.sh
```

- 설치 항목
  - OpenJDK 8
  - Gradle 7.4
  - Docker
  - Docker-compose
- 실행 내용
  - docker | tooliv-network 생성
  - docker | tooliv-volume 생성
  - docker | tooliv-db-volume 생성

## Configure Gradle

Gradle의 bin 디렉토리를 포함하도록 PATH 환경변수를 설정합니다.

gradle.sh 파일을 생성합니다.

```
1 vim ${path}/etc/profile.d/gradle.sh
```

gradle.sh 내부에 아래 내용을 입력합니다.

```
1 export GRADLE_HOME=/opt/gradle/latest
2 export PATH=${GRADLE_HOME}/bin:${PATH}
```

생성한 gradle.sh 파일에 실행 권한을 부여합니다.

```
1 chmod +x ${path}/etc/profile.d/gradle.sh
```

명령어를 입력하여 환경변수를 로딩합니다.

```
1 . ${path}/etc/profile.d/gradle.sh
```

## Install Redis

① TOOLIV 채팅 기능은 빠른 데이터 조회를 위해 Redis를 이용합니다.

아래 명령어를 통해 `redis-container` 를 구동시킵니다.

( 예시 명령어 그대로 입력해서 포트 번호를 동일하게 실행하는 것을 권장합니다. )

```
1 docker run --network tooliv-network -d --name redis-container -p 6380:6380 -dit
```

`redis-container` host 정보를 확인하기 위해 `tooliv-network` 정보를 조회합니다.

```
1 docker inspect tooliv-network
```

```
"Network": "",
},
"ConfigOnly": false,
"Containers": {
  "107323c7d88e14e18e300ece496a73e0327309868fe5ee9b250b16": {
    "Name": "redis-container",
    "EndpointID": "e9a1d27904276d5bd96251889c93936f9d88",
    "MacAddress": "02:42:ac:12:00:02",
    "IPv4Address": "172.18.0.2/16",
    "IPv6Address": ""
  }
}
```

redis-container host 확인 예시

① host 정보는 [Build And Run Project](#) 과정에서 필요한 정보로, 메모해두시기 바랍니다.

# Install OpenVidu

 TOOLIV 화상회 기능은 OpenVidu 오픈소스 프로젝트를 이용합니다.

OpenVidu 설치 위해 /opt 폴더로 이동합니다.

```
1 cd ${path}/opt
```

아래 명령어를 실행하면 OpenVidu가 다운로드되고 설치 스크립트가 동작합니다.

openvidu 폴더에 필요한 모든 파일이 다운로드됩니다.

```
1 curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh
```

openvidu 폴더로 이동합니다.

```
1 cd openvidu
```

OpenVidu 내부적으로도 NGINX를 사용하고 있기 때문에 충돌을 방지하기 위해서 NGINX 실행을 멈춥니다.

```
1 service nginx stop
```

OpenVidu 초기 설정을 위해 환경변수를 다음과 같이 수정합니다.

```
1 vim .env
```

- 환경변수 속성
  - **DOMAIN\_OR\_PUBLIC\_IP** : 도메인네임 또는 Public IP를 작성합니다.
  - **OPENVIDU\_SECRET** : OpenVidu 서버에 연결하기 위해 사용됩니다. (**TOOLIV** 작성)
  - **CERTIFICATE\_TYPE** : letsencrypt로 설정하여 유효한 인증서를 자동 생성합니다.
  - **LESENCRYPT\_EMAIL** : 이메일을 입력합니다.

```
1 # Domain name. If you do not have one, the public IP of the machine.
2 # For example: 198.51.100.1, or openvidu.example.com
3 DOMAIN_OR_PUBLIC_IP={도메인}
4 OPENVIDU_SECRET=TOOLIV
5 CERTIFICATE_TYPE=letsencrypt
6 LETSENCRYPT_EMAIL={이메일}
7 HTTP_PORT=80
8 HTTPS_PORT=443
```



```
# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP= your domain here *

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=TOOLIV

# Certificate type:
# - selfsigned: Self signed certificate. Not recommended for production use.
#               Users will see an ERROR when connected to web page.
# - owncert:    Valid certificate purchased in a Internet services company.
#               Please put the certificates files inside folder ./owncert
#               with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#               required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#               variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL= your email here *

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.
# WARNING: the default port 80 cannot be changed during the first boot
# if you have chosen to deploy with the option CERTIFICATE_TYPE=letsencrypt
HTTP_PORT=80

# Changes the port of all services exposed by OpenVidu.
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=443
```

OpenVidu 초기 설정 화면

속성 값을 모두 입력하였으면 vim 창을 저장 종료합니다. (esc + : wq!)

OpenVidu를 start하여 초기 설정을 완료합니다.

```
1 ./openvidu start
```

실제 **TOOLIV** 서비스에서 사용할 **PORT** 번호를 OpenVidu 환경 변수에 다시 매핑합니다.

```
1 vim .env
```

**HTTP\_PORT** , **HTTPS\_PORT** 2가지 속성 값을 변경합니다.

( 가능한 동일한 설치 환경을 구성하기 위해 예시대로 작성하는 것을 권장합니다. )

```
1 # Domain name. If you do not have one, the public IP of the machine.
2 # For example: 198.51.100.1, or openvidu.example.com
3 DOMAIN_OR_PUBLIC_IP={도메인}
4 OPENVIDU_SECRET=TOOLIV
5 CERTIFICATE_TYPE=letsencrypt
6 LETSENCRYPT_EMAIL={이메일}
7 HTTP_PORT=4442
8 HTTPS_PORT=4443
```

```

DOMAIN_OR_PUBLIC_IP=k6a4021.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=TOOLIV

# Certificate type:
# - selfsigned: Self signed certificate. Not recommended for production use.
#               Users will see an ERROR when connected to web page.
# - owncert:    Valid certificate purchased in a Internet services company.
#               Please put the certificates files inside folder ./owncert
#               with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#               required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#               variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=gganzii1215@gmail.com

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.
# WARNING: the default port 80 cannot be changed during the first boot
# if you have chosen to deploy with the option CERTIFICATE_TYPE=letsencrypt
HTTP_PORT=4442 changed here *

# Changes the port of all services exposed by OpenVidu.
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=4443 changed here *

```

실제 사용할 PORT 번호를 다시 설정해 준 화면

변경을 완료했으면 OpenVidu 설정을 마치고 다시 구동시킵니다.

```
1 ./openvidu restart
```

이제 OpenVidu 설치 및 설정이 끝났습니다.

OpenVidu 작업을 위해 종료했던 NGINX를 다시 구동시키고, 홈 디렉토리로 이동합니다.

```
1 service nginx start
2 cd ${path}/home/ubuntu
```

## Install MySQL and Create Database

TOOLIV Server를 이용하기 위해서는 MySQL 데이터베이스가 필요합니다. 사용자의 환경에 맞게 5.7 버전을 설정하고 컨테이너로 구동시킵니다.

아래 예시는 `tooliv-db-volume` 을 사용하여 `tooliv-db` 라는 이름으로 컨테이너를 구동시키고 `tooliv_db` 스키마를 생성하는 예시 명령어입니다.

```

1 docker run --network tooliv-network -v tooliv-db-volume:/var/lib/mysql --name tooliv-db
2 docker exec -it tooliv-db bash
3 mysql -u root -p
4 패스워드 입력
5 create database if not exists tooliv_db collate utf8mb4_general_ci;

```

① 컨테이너 이름, root 계정 패스워드 및 스키마 이름은 사용자가 직접 설정하면 됩니다.  
추후 환경설정 시 필요한 정보로, 메모해두시기 바랍니다.

## Build And Run Project

clone 받은 client 프로젝트를 Docker Image 빌드합니다.

- Docker Image : tooliv-client:its

```
1 cd ${path}/home/ubuntu/Tooliv/client
2 docker build -t tooliv-client:its .
```

[Install Certbot](#) 과정 중 NGINX에서 설정한 대로 client 컨테이너를 해당 포트 번호로 실행합니다.

가이드대로 진행을 했다면 포트 번호 3000 을 그대로 실행하면 됩니다.

```
1 docker run --network tooliv-network --name tooliv-client -p 3000:3000 -d tooliv-
```

clone 받은 server 프로젝트를 Docker Image 빌드합니다.

- Docker Image : tooliv-server:its

```
1 cd ../server
2 gradle wrapper clean build -x test
3 docker build -t tooliv-server:its .
```

위에서 빌드한 Docker Image를 이용하여 client server 컨테이너를 구동시킵니다.

사진 및 파일을 저장하는 AWS S3 Bucket은 별도의 설정을 하지 않는 경우, Tolliv Bucket을 사용하게 됩니다. 파일을 기업에서 별도로 관리하고 싶은 경우에는 자체 AWS S3 Bucket을 생성하고 `access_key`, `secret_key`, `bucket_name`, `aws_region` 정보를 기입하여 사용할 수 있습니다.

- 환경변수 속성
  - `db_container_name` : tooliv-db (별도로 설정했다면 해당 database 컨테이너명)
  - `db_schema_name` : tooliv\_db (별도로 설정했다면 해당 database명)
  - `database_username` : root (다 계정을 추가했다면 root 혹은 해당 username)
  - `database_password` : 패스워드
  - `aws_accesskey` : S3 Bucket 생성 시 발급된 access key
  - `aws_secret_key` : S3 Bucket 생성 시 발급된 secret key
  - `bucket_name` : S3 Bucket 명
  - `aws_region` : S3 Bucket region (예시: ap-northeast-2)
  - `ssl_key_password` : letsencrypt certbot 을 이용하여 생성한 SSL 비밀번호
  - `redis-container_host` : [Install Redis](#) 과정에 확인한 redis-container host IP
  - `redis-container_port` : 6379 (별도로 설정했다면 해당 port)

자체 S3 Bucket을 사용할 경우

TOOLIV 제공 S3 Bucket을 사용할 경우

```
1 docker container run --network tooliv-network --name tooliv-server
2 -e JAVA_TOOL_OPTIONS="-Dspring.profiles.active=prod"
3 -e "SPRING.DATASOURCE.URL=jdbc:mysql://{db_container_name}/{db_schema_name}"
4 -e "SPRING.DATASOURCE.HIKARI.USERNAME={database_username}"
5 -e "SPRING.DATASOURCE.HIKARI.PASSWORD={database_password}"
6 -e "CLOUD.AWS.CREDENTIALS.ACCESSKEY={aws_access_key}"
7 -e "CLOUD.AWS.CREDENTIALS.SECRETKEY={aws_secret_key}"
8 -e "CLOUD.AWS.S3.BUCKET={bucket_name}"
9 -e "CLOUD.AWS.region.static={aws_region}"
10 -e "SERVER.SSL.KSY-STORE-PASSWORD={ssl_key_password}"
11 -e "SPRING.REDIS.HOST={redis-container_host}"
12 -e "SPRING.REDIS.PORT={redis-container_port}"
13 -p 8443:8443 -d tooliv-server:lbs
```

자체 S3 Bucket을 사용할 경우

TOOLIV 제공 S3 Bucket을 사용할 경우

```
1 docker container run --network tooliv-network --name tooliv-server
2 -e JAVA_TOOL_OPTIONS="-Dspring.profiles.active=prod"
3 -e "SPRING.DATASOURCE.URL=jdbc:mysql://{db_container_name}/{db_schema_name}"
4 -e "SPRING.DATASOURCE.HIKARI.USERNAME={database_username}"
5 -e "SPRING.DATASOURCE.HIKARI.PASSWORD={database_password}"
6 -e "SERVER.SSL.KSY-STORE-PASSWORD={ssl_key_password}"
7 -e "SPRING.REDIS.HOST={redis-container_host}"
8 -e "SPRING.REDIS.PORT={redis-container_port}"
9 -p 8443:8443 -d tooliv-server:lbs
```

## Ready To Use

정상적으로 설치를 완료했다면 관리자 계정으로 접속해서 TOOLIV을 이용할 수 있습니다.

관리자 계정은 최초 1회 자동으로 생성해드리고 있으니, 접속 후 비밀번호를 변경하고 이용 부탁드립니다.

① 관리자 계정 : admin@tooliv.io

비밀번호 : a12345678!@