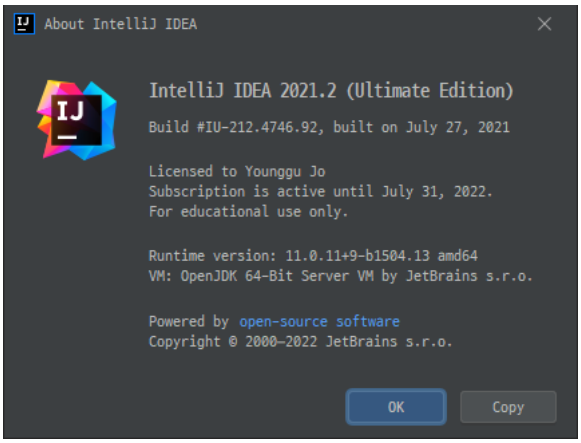
1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

AWS Server 작업 (Ubuntu 20.04.3 LTS)

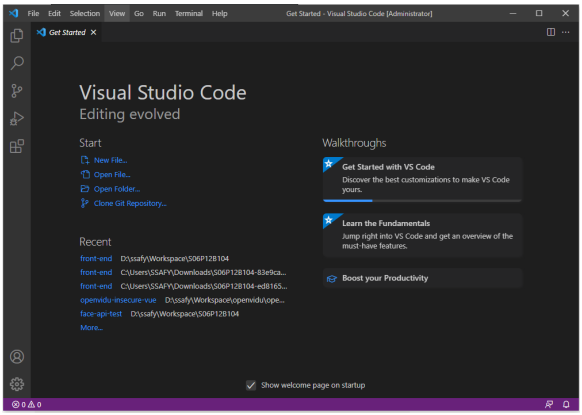
0. 서버 환경 및 설정 & 개발 환경

- Java version
  - openjdk version "1.8.0_312"
  - OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1~20.04-b07)
  - OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
- Web serer
  - nginx version: nginx/1.18.0 (Ubuntu)
- Docker : Docker version 20.10.12, build e91ed57

Back-End 개발 툴 (Intellij)



Front-End 개발 툴 : Microsoft Visual Studio Code



1. AWS nginx 작업

apt update&upgrade

```
# apt update&upgrade
sudo apt upgrade -y
sudo apt update
```

UFW 설치

```
> sudo apt-get update
> sudo apt-get upgrade

> sudo apt-get install ufw
```

UFW 셋팅

```
# 반드시 순서대로 진행
sudo ufw allow 22 # ssh를 위함
sudo ufw enable # 방화벽 설정 (22번 포트를 반드시 연 후 설정)
sudo ufw allow 80 # http를 위함
sudo ufw allow 443 # https를 위함
sudo ufw allow 3306 # database(mysql을 위함)
sudo ufw status # 모든 설정이 되었는지 확인
```

Java 환경변수 설정

```
# java 설치
sudo apt install openjdk-8-jre-headless
# Java version 확인
java -version


#Java 환경변수 설정

# 1. Java 절대 경로 찾기
which java
readlink -f /usr/bin/java

# 알아낸 경로를 통해 환경변수 설정
sudo vim /etc/profile

# vim으로 연 /etc/profiled에 다음 내용을 맨 아래에 삽입&저장
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.amzn2.0.1.x86_64/jre/bin/java
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.jar

# java 환경변수 확인
echo $JAVA_HOME
```

nginx 설치

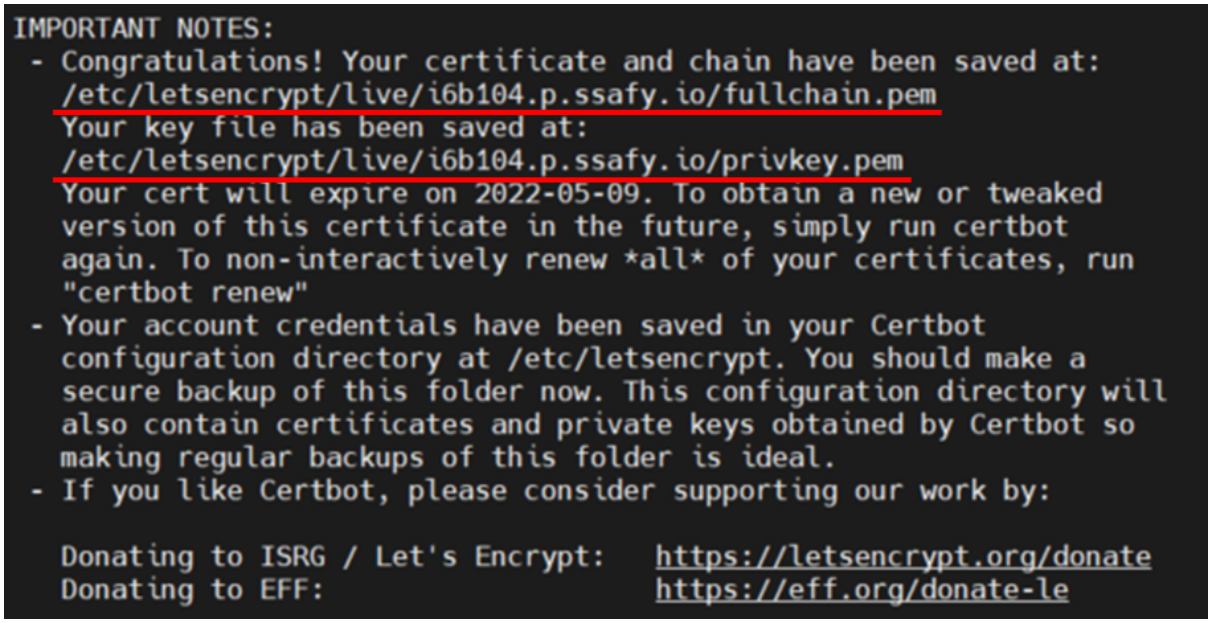
```
# nginx 설치
sudo apt install nginx

# nginx 설치 확인
apt info nginx
```

SSL 인증서 발급

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

```
# nginx 중지
sudo systemctl stop nginx

# 인증서 발급
sudo systemctl stop nginx

# /etc/letsencrypt/live/i6b104.p.ssafy.io-0001 디렉터리 이동 후
# fullchain.pem, privkey.pem 생겼는지 확인
cd /etc/letsencrypt/live/i6b104.p.ssafy.io-0001
ls -l
```

인증서 정상 발급 후 보이는 화면



nginx 설정

```
# nginx 기본 설정 (경로 변경을 위해 편집)
sudo vi /etc/nginx/sites-available/default

# 다음 내용대로 설정&저장
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        server_name i6b104.p.ssafy.io;

        return 301 https://$server_name$request_uri;

        index index.html index.htm;
}

server {
        listen 443 ssl;
        listen [::]:443 ssl;

        server_name i6b104.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/i6b104.p.ssafy.io-0001/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i6b104.p.ssafy.io-0001/privkey.pem;

        root /home/ubuntu/dist;
        index index.html;

        location / {
                try_files $uri $uri/ /index.html;
        }

        location /api {
                #proxy_pass http://localhost:8080;
                proxy_pass http://i6b104.p.ssafy.io:8080;
                proxy_redirect off;
                charset utf-8;
```

```
                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 X-NginX-Proxy true;
        }
}

# nginx enable 설정
sudo systemctl -l enable nginx

# nginx 실행 & 재실행
sudo service nginx start
sudo service nginx restart

# nginx 상태 확인하기
systemctl status nginx

# nginx 오류 확인
sudo nginx -t

# net-tools 설치
sudo apt install net-tools

# 열려있는 모든 포트 확인
netstat -ano
```

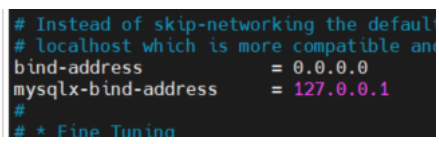
2. Database 설정 (Mysql)

Mysql 계정&DB 생성

```
# MySQL 설치
sudo apt update
sudo apt install mysql-server

# Mysql 접속
sudo mysql -u root -p mysql
# 이후 Enter password에 mysql 비밀번호 입력 후 일련의 작업 실행

# 특정 사용자 생성
create user 'azit'@'%' identified by '<Mysql_password>';
# 특정 사용자에게 권한 부여 (%:모든 곳에서 접속 허용)
grant all privileges on *.* to azit@'%';
# 변경사항 저장
flush privileges;

# Database 계정 권한 보기
show grants for azit;

# database 생성
create database if not exists `azit` collate utf8mb4_general_ci;

# mysql 종료
exit
```

Mysql 설정

```
# 해당 경로로 이동
cd /etc/mysql/mysql.conf.d
# mysql.config 파일을 열어 IP주소를 변경
sudo vi mysqld.cnf

# bind-address 주소 변경&저장 (127.0.0.1 -> 0.0.0.0)
bind-address = 0.0.0.0
```



배포 (back-end/front-end/openvidu)

1. Back-End

`application.properties` 파일을 다음과 같이 설정

```
# server 관련 설정
server.port=8080

# spring JPA 설정
spring.jpa.hibernate.ddl-auto=create
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.database=mysql

# database 설정
spring.datasource.url=jdbc:mysql://localhost:3306/azit?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.datasource.username=azit
spring.datasource.password=<database_password>
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver`
```

`back-end/Azit` 디렉토리에서 아래 명령어 실행

```
.\gradlew clean build
# back-end/Azit/build/libs에 Azit-0.0.1-SNAPSHOT.jar 파일 생성됨!
```

서버에 파일 전송

```
# scp 명령어로 aws EC2 서버에 파일 업로드
sudo scp -i i6b104T.pem Azit-0.0.1-SNAPSHOT.jar \
ubuntu@i6b104.p.ssafy.io:/home/ubuntu
```

AWS server 접속 후 back-end 서버 실행

```
# Backend 실행
java -jar Azit-0.0.1-SNAPSHOT.jar
# 최초실행후 종료
```

실행&종료 후,

`application.properties` 파일을 다음과 같이 설정

```
# server 관련 설정
server.port=8080

# spring JPA 설정
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.database=mysql

# database 설정
spring.datasource.url=jdbc:mysql://localhost:3306/azit?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.datasource.username=azit
spring.datasource.password=<database_password>
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

`back-end/Azit` 디렉토리에서 아래 명령어 실행

```
.\gradlew clean build
# back-end/Azit/build/libs에 Azit-0.0.1-SNAPSHOT.jar 파일 생성됨!
```

서버에 파일 전송

```
# scp 명령어로 aws EC2 서버에 파일 업로드
sudo scp -i i6b104T.pem Azit-0.0.1-SNAPSHOT.jar \
ubuntu@i6b104.p.ssafy.io:/home/ubuntu
```

```
# nohub을 통한 백그라운드 실행 (무중단 배포)
nohup java -jar Azit-0.0.1-SNAPSHOT.jar &
# nohub을 통한 백그라운드 실행 (무중단 배포) + log 저장하지 않기
nohup java -jar Azit-0.0.1-SNAPSHOT.jar & > /dev/null

# nohub으로 실행된 백그라운드 프로세스 확인
jobs

# 백그라운드로 실행된 process 검색
ps -ef | grep <프로세스명>
# Azit(back-end) 프로세스 찾기
ps -ef | grep Azit
ps -ef|grep Azit|awk '{print $2}'
# 해당 프로세스의 pid 값으로 프로세스 kill
kill -9 <pid>

#ps 결과물로 나온 프로세스 죽이기 (!!!정확히 검색해야 함!!!)
kill -9 `ps -ef|grep Azit|awk '{print $2}'`


# 브라우저에서 정상 작동하는지 확인
```

2. Front-End

`front-end` 디렉토리에서 아래 명령어 실행

```
# 필요한 package 설치
npm install

# front-end 빌드
npm run build
# ../back-end/Azit/src/main/resources 경로에 dist 디렉토리 생성됨
# 생성된 dist 파일을 wsl등의 터미널을 통해 서버에 전송
sudo scp -ri i6b104T.pem dist ubuntu@i6b104.p.ssafy.io:/home/ubuntu
```

!특이사항!

Windows 환경에서 `npm run build` 실패 시, C++ workload를 포함한 Visual Studio를 설치해야 함.

3. WebRTC_Openvidu

`Docker` 설치

```
$ sudo apt-get update
$ sudo apt-get -y install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

`Docker-compose` 설치

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/d
$ sudo chmod +x /usr/local/bin/docker-compose
```

`AWS EC2` 포트 열기

```
$ sudo su
################################
ufw allow ssh
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/udp
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
ufw enable
```

`OpenVidu` 설치

```
$ sudo su
$ cd /opt
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

`/opt/openvidu` 디렉토리에서 아래 명령어 실행

```
$ cd /opt/openvidu
$ sudo vi .env
```

`.env` 파일에서 환경 설정

```
# SSAFY에서 제공한 서버 도메인 (예: 서울1반1팀 - i6a101.p.ssafy.io)
DOMAIN_OR_PUBLIC_IP=i6b104.p.ssafy.io

# Openvidu 실행 시 사용할 비밀번호
OPENVIDU_SECRET=MY_SECRET

# 자동으로 SSL 인증서 생성
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL=<ssl 인증서 발급 시 사용한 이메일>

# letsencrypt 방식이면 주석 처리 풀어줘야함. (사용할 포트 작성)
HTTP_PORT=8445
HTTPS_PORT=4443

# 녹화 기능 사용을 위해 true로 변경
OPENVIDU_RECORDING=true
```

`OpenVidu` 서버 실행

```
$ ./openvidu start
```