SSAFISH

서버 환경 설정과 프로젝트 배포

공통 프로젝트 (2023.07.10 ~ 2023.08.18)

웹기술(WebRTC)

팀장 : 황유성

팀원 : 신세영 김경주 정재범 홍진환 홍정현

담당 컨설턴트 : 박세영

개요

1. 서버환경 구성하기

1-1. EC2 서버 접속

1-2. Docker 설치 및 설정

1-3. MySQL 설치 및 설정

1-4. Openvidu 설치

1-5. Nginx 설치 및 설정

1-6. 기본 설정(apt, 방화벽,net…)

1-7. 카드 이미지 서버에 저장

2. Git clone

3. 수동 배포

3-1. 프론트

3-2. 백엔드

4. 유의 사항

1. 서버 환경 구성하기

[준비사항]

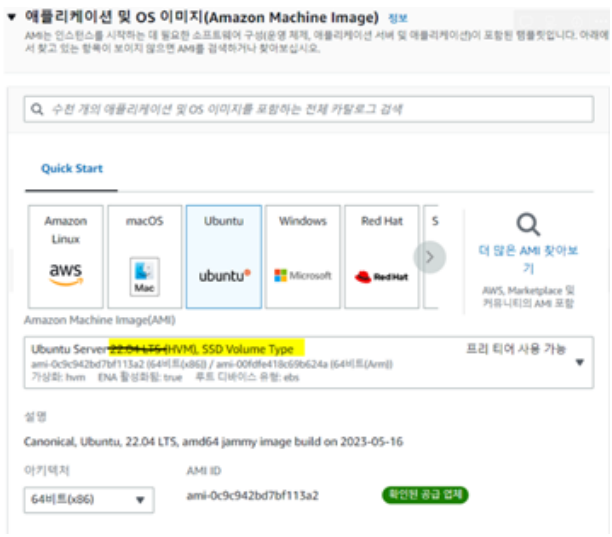
ec2 서버를 사용하기 위해선 **AWS** 회원 가입후 EC2 인스턴스를 생성해야한다. 여기선 간략한 가이드만 제공하므로 다른 게시글을 참고하도록 한다.

(참고로 프로티어에서 사용할수 있는 서버는 성능 문제로 openvidu를 올릴 수 없다.)

[*개발시와 동일한 버전 안내]

Tip. ec2 인스턴스 생성을 위한 간략한 가이드

1. 인스턴스를 생성하고 서버를 선택한다. → [ubuntu 20.04 LTS]



2. .pem 키생성 (이후 서버에 접속할 때마다 사용하므로 잘 저장해둔다.)

3. 보안 그룹 설정(포트를 열어 두어야한다.)

1-1. EC2 서버 접속

- 생성한 인스턴스를 실행시키고, 선택하면 위와 같은 화면을 볼 수 있다.
- 활성화된 “연결” 버튼을 **선택하면 ssh를 이용해 서버에 접속할수 있는 명령어를 복사** 할수 있다.
- **이를 .pem 키를 저장한 파일위치에서 cmd창을 실행해 입력하면** 서버에 접속 할 수 있다.

1-2. Docker 설치 및 설정

1. 기존에 깔려 있을지 모르는 도커를 미리 제거한다.

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

2. 사전 준비

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

sudo apt install gnome-terminal

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3. 도커 설치 및 테스트

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo docker run hello-world
```

4. 도커 명령어 실행시, sudo 명령어를 제외하기 위한 설정

```
sudo groupadd docker
sudo usermod -aG docker root
sudo service docker restart
```

aws 에서 docker 를 사용하기 위한 기본명령어

- 실행중인 컨테이너 목록을 출력

```
docker ps
```

- 실행중이거나 종료된 컨테이너 목록을 출력

```
docker ps -a
```

- AWS에 pull 된 모든 이미지 목록을 출력

```
docker images
```

- 실행중인 컨테이너의 로그를 확인

```
docker logs [이미지 명 or 이미지 ID]
```

1-3. MySQL 설치 및 계정 설정

AWS 서버에 접속한 상태에서 다음 명령어를 차례로 실행한다.

1. MySQL 설치 및 버전확인

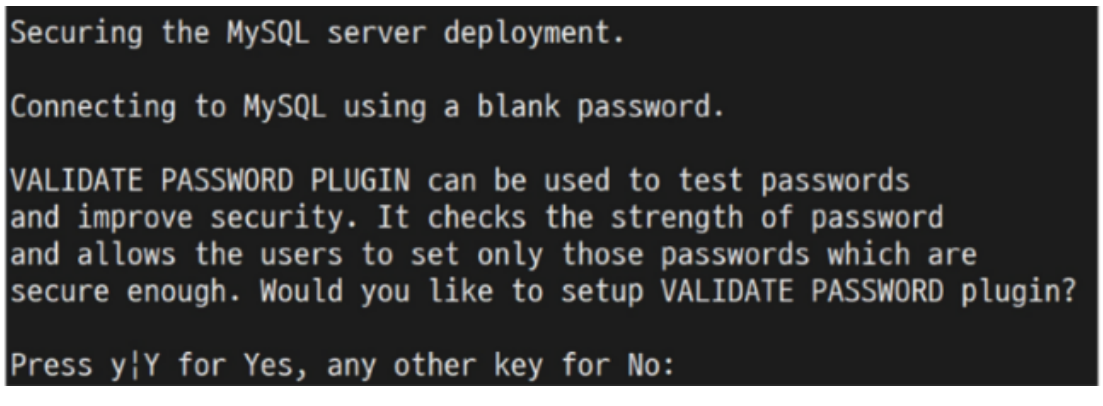
```
sudo apt update
sudp apt install mysql-server
mysql --version
```

2. 보안 설정

```
sudo mysql_secure_installation
```

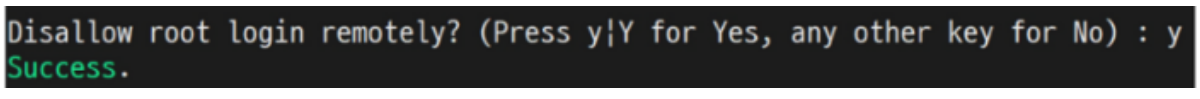
[기타 옵션 설정]

- 비밀번호 유효성 검사를 위한 플러그인 설치 여부 → Enter로 넘긴다.

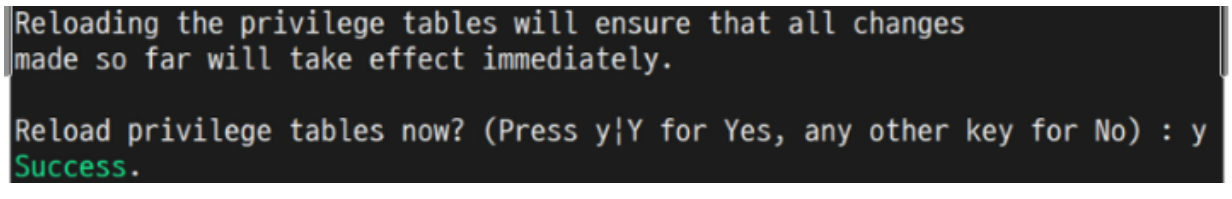


- 임의 유저 제거를 물으면 y로 제거한다.

- 원격 접속을 허용하지 않는다.



- 변경 설정을 즉시 적용할 것인지.



3. 권한 부여

- aws 에서 my sql에 접속한다.

```
sudo mysql
```

- 새로운 관리자 권한을 부여한다.(이후 로컬에서 접속하기 위해 사용한다.)

```
create user 'e202'@'%' indentified by 'ssafish';
grant all privileges on *.* to 'e202'@'%';
```

- 생성한 관리자 권한으로 접속(해당 명령어 입력 후 비밀번호(ssafish)를 입력한다.입력 내용은 보이지 않으므로 주의해서 입력하도록 한다.)

```
sudo mysql -u e202 -p
```

- mysql 에서 나오기

```
exit
```

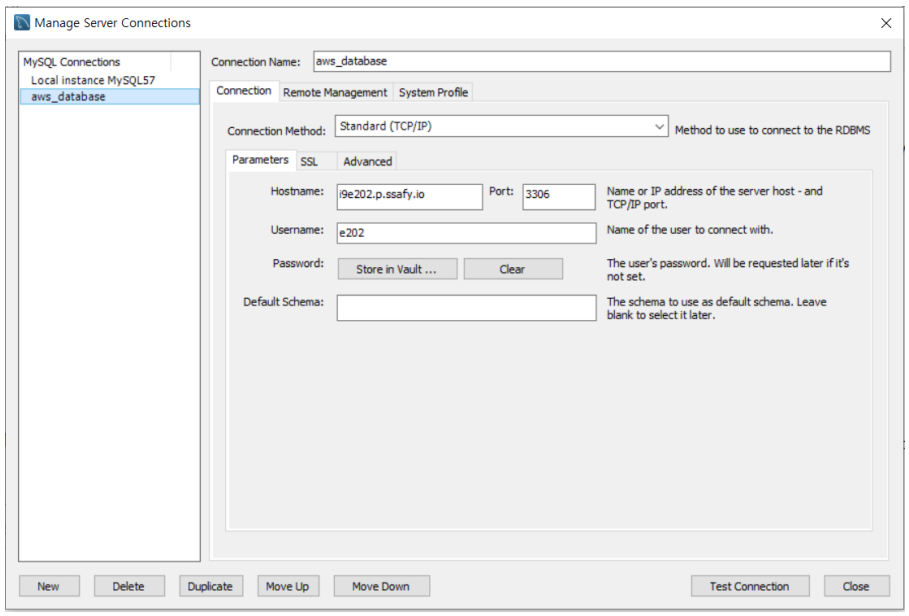
- MySQL에 접속할수 있는 IP주소 (Listen IP)를 설정한다.

```
cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep bind
```

해당 명령어 입력 후 bind-address 값을 개발 PC의 IP주소로 변경한다.

임시로 0.0.0.0 으로 설정해 자유롭게 사용할 수 있으나 특정 IP만 허용하는 것을 추천한다.

4. 로컬의 MySQL Workbench에서 서버의 mysql에 접속하도록 설정



(3번 권한 부여 시 설정한 내용)

Hostname	aws의 퍼블릭 IPv4주소
Username	e202
Password	ssafish

1-4. Openvidu 설치

[참고용 공식 링크]

https://docs.openvidu.io/en/2.28.0/deployment/ce/on-premises/#close-ports-to-avoid-external-attacks

1. 서버의 해당 위치로 이동하여, Openvidu 서버 설치

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

2. 변수 세팅

opt/openvidu/.env 파일에서 다음과 같이 설정을 변경해 준다.

```
DOMAIN_OR_PUBLIC_IP=i9e202.p.ssafy.io #도메인 설정
OPENVIDU_SECRET=openvidu # 비밀번호 설정
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL= # 사용자의 이메일
HTTP_PORT=8081
HTTPS_PORT=4443
```

3. openvidu 서버 실행 : /opt에서 아래 명령어를 실행한다.

```
./openvidu start
```

혹은 재시작 하거나 ,정지시키고 싶을 땐 아래와 같다.

```
./openvidu restart
./openvidu stop
```

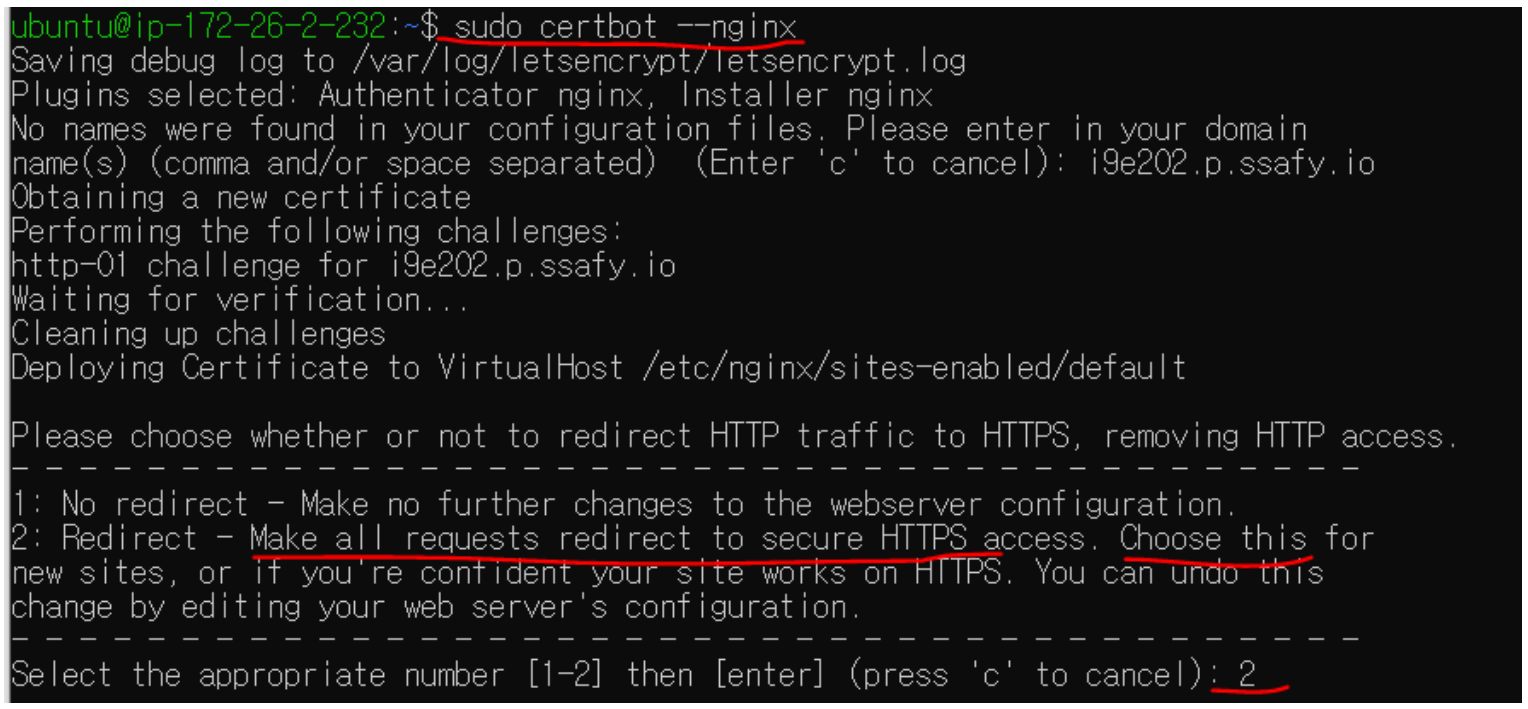
1-5. Nginx 설치 및 설정

1. nginx 설치

```
//설치
sudo apt-get install nginx
//환경 확인
sudo systemctl status
```

2. certbot 설치

```
sudo apt-get install letsencrypt
sudo apt-get install certbot python3-certbot-nginx
//certbot 동작
sudo certbot --nginx
```



3. nginx 환경설정

```
sudo vi /etc/nginx/sites-availavble/default
```

4. 설정 내용

```
server {

        location /api {
        proxy_pass http://localhost:8080;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location / {
                proxy_pass http://localhost:5173;
        }

        location /ssafish {

            proxy_pass http://localhost:8080;

            proxy_read_timeout 3600s;
            proxy_send_timeout 3600s;

            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;

        }
        location /main_images/ {
                alias /home/ubuntu/ssafish/cardMainImage/;
        }

        location /sub_images/ {
                alias /home/ubuntu/ssafish/cardSubImage/;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
```

```
    ssl_certificate /etc/letsencrypt/live/i9e202.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i9e202.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = i9e202.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
       listen 80 ;
       listen [::]:80 ;
    server_name i9e202.p.ssafy.io;
    return 404; # managed by Certbot
}
```

5. nginx 재실행

```
sudo service nginx restart
```

1-6. 기본 설정

1. 포트 사용 현황을 확인하기 위한 네트워크 툴설치[필수 아님]

```
// 설치
sudo apt install net-tools
//실행방법, 둘중하나를 선택하면된다.
netstat -tnpl
sudo netstat -npl
```

2. 방화벽설정(만약 방화벽의 포트를 열어 줬음에도 포트 연결관련 오류가 발생하는 경우 EC2 자체의 보안 설정을 확인한다.)

```
//설치
sudo apt install ufw
//설정
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 3478/tcp
sudo ufw allow 3478/udp
sudo ufw allow 40000:57000/tcp
sudo ufw allow 40000:57000/udp
sudo ufw allow 57001:65535/tcp
sudo ufw allow 57001:65535/udp
sudo ufw allow 3306
sudo ufw allow 8081
sudo ufw allow 8080
sudo ufw allow 4443
sudo ufw allow 5173
sudo ufw allow 9090
sudo ufw allow 5001
sudo ufw allow 5000
sudo ufw enable
```

1-7. 카드 이미지 서버에 저장

1. 경로는 자신에게 맞게 적절히 변경한다.

```
scp -i "D:\...\ .pem" -r D:\E202\cardMainImage ubuntu@i9e202.p.ssafyio:~/ssafish/cardMainImage
```

2. 카드를 서버에서 읽고 쓰기 위해선 서버에서 각 폴더에 적절한 권한을 설정하고

(가장 간단한 권한 설정법은 chmod 777 [파일/폴더명]),

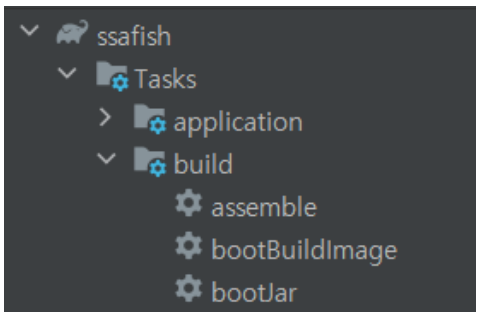
이후 배포에서 설명 하듯이 컨테이너에 바인드 바운트 해주어야 한다.

2. Git clone

3. 수동 배포

3-1. 백엔드

1. gradle의 bootJar을 클릭해 Jar파일을 생성한다.(IntelliJ우측 상단),



그러면 build/libs내부 jar파일이 생성된다.

2. 터미널에서 Dockerfile이 있는 위치로 이동하여 아래 명령어를 실행하면 로컬에 이미지가 생성된다.

```
docker build -t ${자신의 저장소명}/ssafish .
```

3. 이를 DockerHub에 push한다.

```
docker login
docker push ${자신의 저장소명}/ssafish:latest
```

4. DockerHub에서 백엔드 이미지를 아래 명령어를 통해 pull받아 실행한다.

```
 //백엔드
docker run -d -t -p 8080:5001  -v /home/ubuntu/ssafish:/home/ssafish --rm --name ssafish ${자신의저장소명}/ssafish:latest
```

이때카드 이미지 파일을 서버에서 읽고 써야 하므로 -v /home/ubuntu/ssafish:/home/ssafish 로 바인드 마운트 되었다.

3-2. 프론트

1. npm run build 명령어를 통해 프로젝트를 빌드 한다.

```
npm run build
```

2. 터미널에서 Dockerfile이 있는 위치로 이동하여 아래 명령어를 실행하면 로컬에 이미지가 생성된다.

```
docker build -t ${자신의 저장소명}/ssafish-fe .
```

3. 이를 DockerHub에 push한다.

```
docker login
docker push ${자신의 저장소명}/ssafish-fe:latest
```

4. DockerHub에서 프론트 이미지를 아래 명령어를 통해 pull받아 실행한다.

```
//프론트
docker run -d -t -p 5173:5173 --rm --name ssafish-fe ${자신의저장소명}/ssafish-fe:latest이때 백엔드 이미지의 경우 카드 이미지 파일을 서버에서 읽고 써야 하므로  -v /home/ubuntu/ssaf.
```

4. 유의 사항

- 백엔드 코드 실행 시 e2_imgs 파일은 존재하지 않으므로 읽어올 수 없다.