🔵 배포 정리 문서

🌐 배포 설정

프론트엔드 빌드 및 배포

1. **git clone**

```
git clone <https://lab.ssafy.com/s08-webmobile1-sub2/S08P12A804.git>
```

2. **frontend 폴더로 이동**

```
cd ./S08P12A804/frontend
```

3. **Dockerfile 작성**

```
# node 이미지 사용
FROM node:alpine as builder

WORKDIR /usr/src/app

COPY package.json .

# 모듈 설치
RUN npm install

COPY ./ ./

# 빌드 파일 생성
RUN npm run build

# nginx 이미지 사용
FROM nginx:stable-alpine

RUN mkdir /app

WORKDIR /app

RUN mkdir ./build
```

```
# 빌드한 파일을 ./build로 복사
COPY --from=builder /usr/src/app/build/ ./build

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 3000

# 컨테이너 실행시 서버 실행
CMD ["nginx", "-g", "daemon off;"]
```

4. 도커 이미지 생성

```
sudo docker build -t {도커 이미지 이름} .
```

5. 도커 컨테이너 생성

```
sudo docker run -p {외부에서 연결할 포트번호}:3000 --name {도커 컨테이너 이름} -d -v {EC2 서버 폴더 경로}:{생성할 도커 컨테이너의 폴더 경로} {도커 이미지 이름}
```

백엔드 빌드 및 배포

1. git clone

```
git clone <https://lab.ssafy.com/s08-webmobile1-sub2/S08P12A804.git>
```

2. backend 폴더로 이동

```
cd ./S08P12A804/backend
```

3. Dockerfile 작성

```
# openjdk 이미지 사용
FROM openjdk:8-jdk-alpine

EXPOSE 8040

ARG JAR_FILE=build/libs/kkini-0.0.1-SNAPSHOT.jar
```

```
COPY ${JAR_FILE} app.jar

# 컨테이너 실행시 jar 파일 실행
ENTRYPOINT ["java","-jar","/app.jar"]

ENV TZ=Asia/Seoul
```

4. **빌드**

```
./gradlew -x test clean build
```

5. **도커 이미지 생성**

```
sudo docker build -t {도커 이미지 이름} .
```

6. **도커 컨테이너 생성**

```
sudo docker run -p {외부에서 연결할 포트번호}:8040 --name {도커 컨테이너 이름} -d -v {EC2 서버 폴더 경로}:{생성할 도커 컨테이너의 폴더 경로} {도커 이미지 이름}
```

---

nginx.conf 파일

```
server {
    listen 80;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

nginx 설정

1. **nginx 설치**

```
sudo apt-get install nginx
```

2. **/etc/nginx/sites-available로 이동후 default 파일 수정 또는 새로운 이름의 파일 생성**

```
server {
        # 프론트 연결
        location /{
                proxy_pass <http://localhost>:{프론트 포트 번호};
        }

        # 백엔드 연결
        location /api {
                proxy_pass <http://localhost>:{백엔드 포트 번호}/api;
        }
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/[도메인 주소]/fullchain.pem; # managed by Certb
ot
    ssl_certificate_key /etc/letsencrypt/live/[도메인 주소]/privkey.pem; # managed by Cer
tbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    # 도메인 이름을 입력
    if ($host = [도메인 주소]) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name [도메인 주소];
    return 404; # managed by Certbot
}
```

3. nginx 실행

```
sudo service nginx start
```

OpenVidu 배포

1. root 권한 얻기

```
sudo su
```

2. openvidu 설치 위치인 /opt로 이동

```
cd /opt
```

3. openvidu 설치

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

4. openvidu 폴더로 이동

```
cd /opt/openvidu
```

5. openvidu 설정 변경

```
$ nano .env

# OpenVidu configuration
DOMAIN_OR_PUBLIC_IP={도메인 주소}

OPENVIDU_SECRET={비밀키}

# Certificate type
CERTIFICATE_TYPE=letsencrypt

# 인증서 타입이 letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL={이메일 주소 ex) ssafy@ssafy.com}

# HTTP port
HTTP_PORT={연결할 포트 번호}

# HTTPS port
HTTPS_PORT={연결할 포트 번호}
```

OpenVidu 실행

```
./openvidu start
```