



포팅메뉴얼

서울 4반 3팀



포팅메뉴얼

기술스택

개발환경

OS

- window 10
- Ubuntu 20.04 LTS

IDE

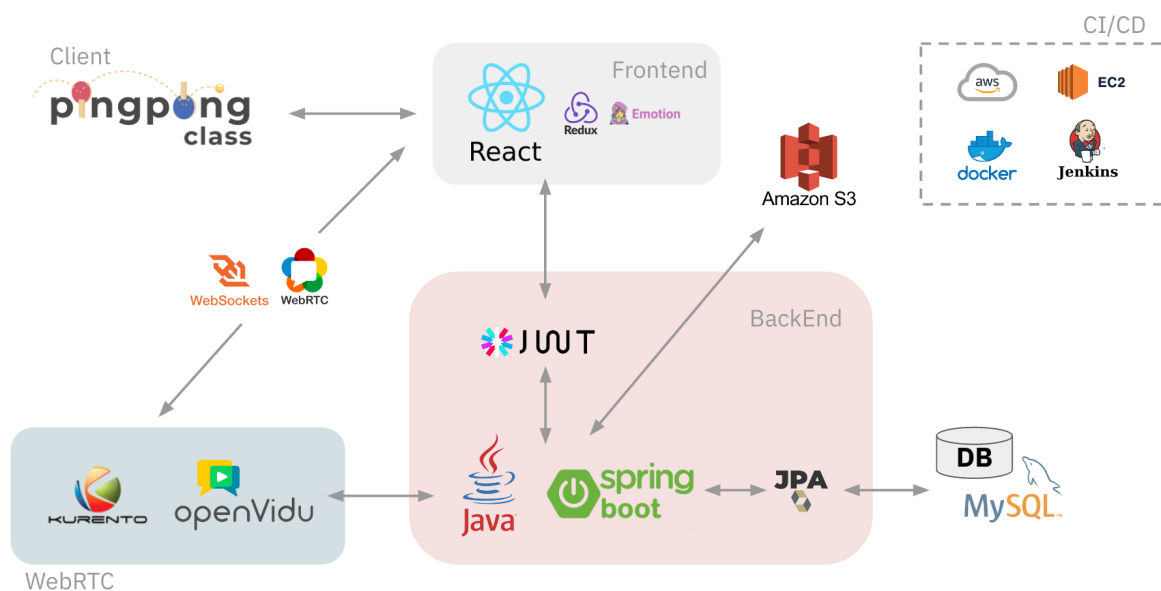
- IntelliJ IDE 2022.1.3
- Visual Studio Code 1.70

Datebase

- MySQL workbench 8.0.28

Docker

- Docker 20.10.12



EC2 기본

Docker 설치

```
# 오래된 버전 삭제하기
sudo apt-get remove docker docker-engine docker.io containerd runc

# repository 설정하기
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# Docker의 Official GPG Key 를 등록
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# stable repository 를 등록
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

# Docker Engine 설치하기
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io

# 설치확인
docker --version
```

Jenkins 설치

- Docker로 Jenkins 설치

```
docker run -d -p 8064:8080 -p 50000:50000 -v \
/var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock \
--name jenkins -u root jenkins/jenkins:lts-jdk11
```

jenkins 컨테이너 안에 docker 설치

- 컨테이너 형태로 설치된 젠킨스 안에서 docker 명령어를 실행하기 위해서 docker를 설치
- 설치를 위해 먼저 컨테이너 안으로 접근, 쉘을 실행.

```
docker exec -it jenkins bash
```

- 도커 설치 → 위의 도커 설치 과정 참고

Docker Image pull

- mysql 5.7

```
docker pull mysql:5.7
```

- node:16-alpine

```
docker pull node:16-alpine
```

- openjdk:8-jre-alpine

```
docker pull openjdk:8-jre-alpine
```

DB

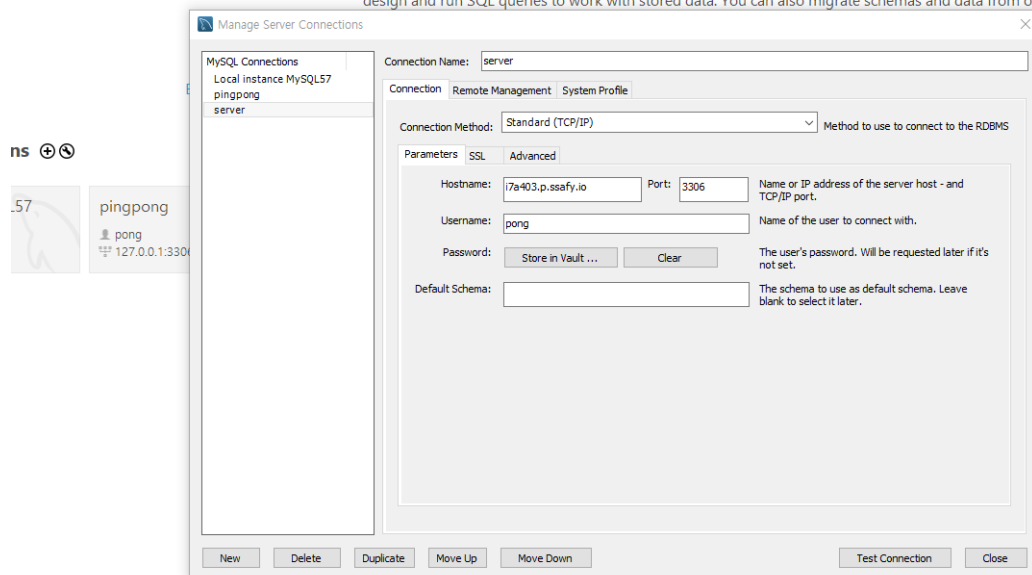
mysql 컨테이너 실행

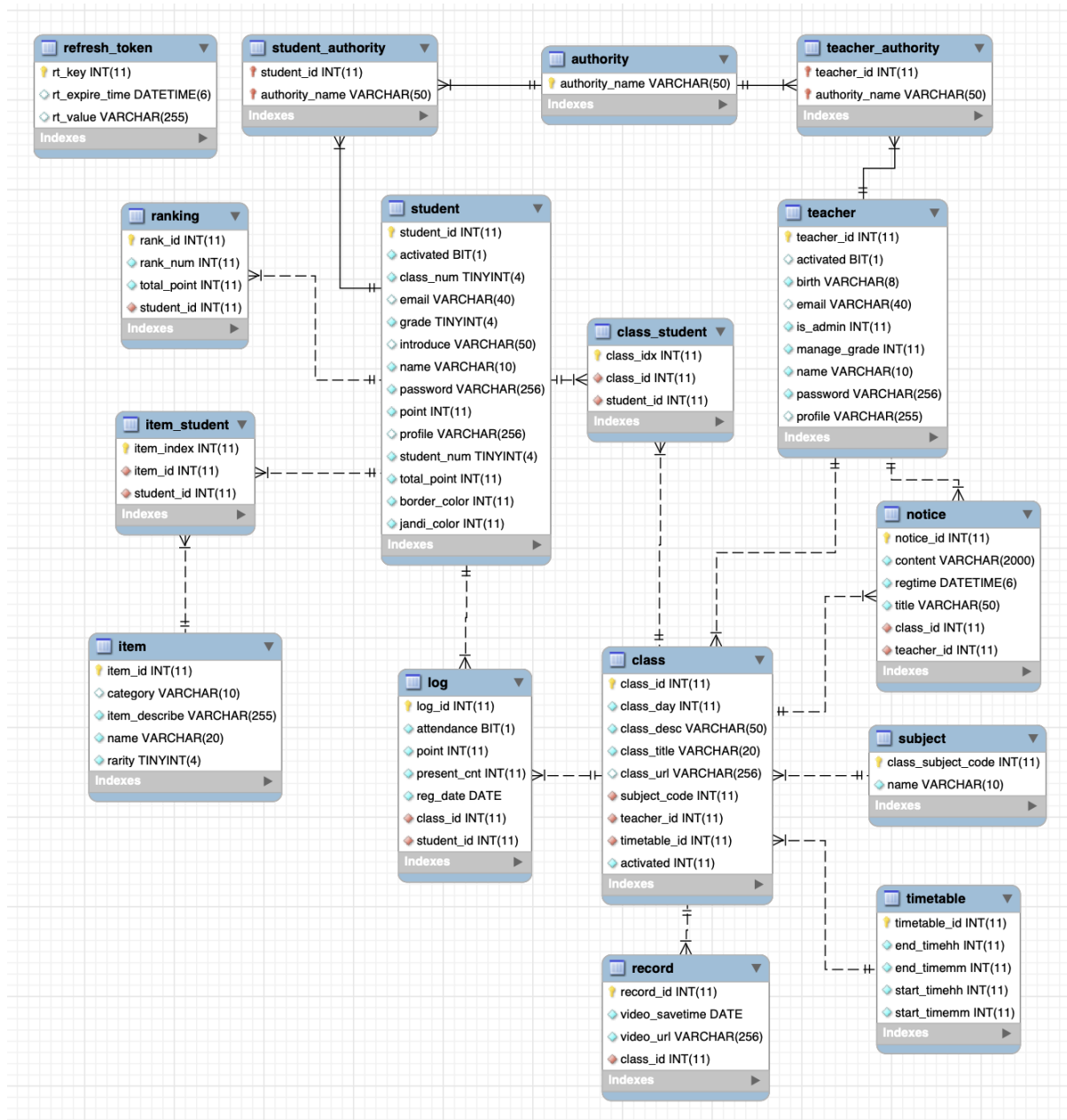
```
docker run --name mysql-pingpong -e MYSQL_ROOT_PASSWORD=ssafy  
-e MYSQL_DATABASE=pingpong -e MYSQL_USER=pong  
-e MYSQL_PASSWORD=pingpong403 -d -p3306:3306 mysql:5.7
```

- workbench에서 연결 후 DB 관리 가능

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other





S3

- 아마존 서버 컴퓨터의 저장공간만 빌리는 서비스. 파일 업로드, 다운로드가 가능하다.
- AWS 계정 필요

IAM - 사용자

IAM > 사용자

사용자 (1) 정보

IAM 사용자는 계정에서 AWS와 상호 작용하는 데 사용되는 장기 자격 증명을 가진 자격 증명입니다.

사용자 이름 또는 액세스 키로 사용자 찾기

사용자 추가

- 사용자 추가 클릭
- 사용자 이름 작성
- AWS 액세스 유형 선택 → 액세스 키 선택 시 api 로 접근 가능한 key 발급

IAM > 사용자

사용자 (1) 정보

IAM 사용자는 계정에서 AWS와 상호 작용하는 데 사용되는 장기 자격 증명을 가진 자격 증명입니다.

🔄 삭제 사용자 추가

🔍 사용자 이름 또는 액세스 키로 사용자 찾기

< 1 > ⓘ

<input type="checkbox"/>	사용자 이름	그룹	마지막 활동	MFA	암호 수명	활성 키 수명
<input type="checkbox"/>	pingpong	없음	✓ 어제	없음	없음	✓ 14일 전

사용자 > pingpong

요약 사용자 삭제 ⓘ

사용자 ARN: am:aws:iam::783226195545:user/pingpong ⓘ
경로: /
생성 시간: 2022-08-03 14:58 UTC+0900

권한 그룹 태그 보안 자격 증명 액세스 관리자

▼ Permissions policies (1 정책이 적용됨)

권한 추가 ⓘ 온라인 정책 추가

정책 이름	정책 유형
작업 연결	
▶ AmazonS3FullAccess	AWS 관리형 정책

▶ Permissions boundary (not set)

▼ CloudTrail 이벤트를 기반으로 정책 생성

이 사용자에 대한 액세스 활동을 기반으로 새 정책을 생성한 다음, 이를 사용자 지정하고 생성하여 이 역할에 연결할 수 있습니다. AWS는 CloudTrail 이벤트를 사용하여 사용된 서비스 및 작업을 식별하고 정책을 생성합니다. 자세히 알아보기 ⓘ

Share your **feedback** and help us improve the policy generation experience.

정책 생성

지난 7일 동안 정책 생성 요청이 없습니다.

요약 사용자 삭제 ⓘ

사용자 ARN: am:aws:iam::783226195545:user/pingpong ⓘ
경로: /
생성 시간: 2022-08-03 14:58 UTC+0900

권한 그룹 태그 보안 자격 증명 액세스 관리자

로그인 자격 증명

요약 • 사용자에게 콘솔 관리 액세스 권한이 없습니다.

콘솔 비밀번호 비활성 | 관리

활성된 MFA 디바이스 할당되지 않음 | 관리

서명 인증서 없음

액세스 키

액세스 키를 사용하여 AWS CLI, PowerShell 등 도구, AWS SDK 또는 직접 AWS API 호출을 통해 AWS를 프로그래밍 방식으로 호출합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다.

보안을 위해 비밀 키를 다른 사람과 공유해서는 안 됩니다. 또한 키를 자주 교체하는 것이 좋습니다.

비밀 키는 생성 시에만 보거나 다운로드할 수 있습니다. 기존 비밀 키를 잘못 보관한 경우 새 액세스 키를 생성하십시오. 자세히 알아보기

액세스 키 만들기

액세스 키 ID	생성 완료	마지막 사용	상태
AKIA3MW7LJZM64XEIT4	2022-08-03 14:58 UTC+0900	ap-northeast-2에서 2022-08-16 18:38 UTC+0900에 s3 사용	(활성) 비활성화

사용자 ARN: am:aws:iam::783226195545:user/pingpong

경로: /

생성 시간: 2022-08-03 14:58 UTC+0900

권한 그룹 태그 보안 자격 증명 액세스 관리자

액세스 관리자는 해당 사용자에게 부여된 서비스 권한과 해당 서비스에 마지막으로 액세스한 시간을 표시합니다. 이 정보를 사용하여 정책을 개정할 수 있습니다. [자세히 알아보기](#)

허용된 서비스 (2)

Access Advisor는 서비스 및 EC2, IAM, Lambda, S3 관리 작업에 대한 활동을 보고합니다. 작업을 보려면 목록에서 서비스 이름을 선택합니다. 최근 서비스 활동은 일반적으로 4시간 내에 표시됩니다. 지난 400일간의 서비스 활동이 보고됩니다. [자세히 알아보기](#)

마지막으로 액세스한 정보를 EC2, IAM, Lambda, S3 관리 작업에 사용할 수 있습니다.

검색 필터 없음

서비스	정책 부여 권한	마지막 액세스 날짜
Amazon S3	AmazonS3FullAccess	어제
Amazon S3 Object Lambda	AmazonS3FullAccess	추적 기간에 액세스되지 않음

S3 - bucket

- 버킷 만들기

Amazon S3 > 버킷

계정 스냅샷
Storage Lens는 스토리지 사용량 및 활동 추세에 대한 가시성을 제공합니다. [자세히 알아보기](#)

버킷 (1) Info

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

이름으로 버킷 찾기

이름	AWS 리전	액세스	생성 날짜
test-ppc-bucket	아시아 태평양(서울) ap-northeast-2	퍼블릭	2022. 8. 3. pm 2:32:37 PM KST

test-ppc-bucket Info

퍼블릭 액세스 가능

객체 속성 권한 지표 관리 액세스 지점

버킷 개요

AWS 리전 아시아 태평양(서울) ap-northeast-2	Amazon 리소스 이름(ARN) arn:aws:s3:::test-ppc-bucket	생성 날짜 2022. 8. 3. pm 2:32:37 PM KST
--------------------------------------	--	--

springboot 설정

aws.yml

```
#S3
cloud:
  aws:
    credentials:
      accessKey: AKIA3MW7LJZM64XEIT4
      secretKey: LbAv7cIGrMajUdeUvodq75zX1/j2exRLLnbTmo5N
    s3:
      bucket: test-ppc-bucket
      region:
        static: ap-northeast-2
      stack:
        auto: false

spring:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
```

Backend - 젠킨스 자동배포

application.yml → DB연결

```
datasource:
  driver-class-name: com.mysql.cj.jdbc.Driver
  hikari:
    password: pingpong403
    username: pong
    url: jdbc:mysql://mysql-pingpong:3306/pingpong?useUnicode=true&character
```

* URL 작성 주의하기

```
url: jdbc:mysql://mysql-pingpong:3306/pingpong?
```

백엔드 컨테이너 빌드 시 link로 db 컨테이너와 직접 연결 안해도 됨.
그럴땐 --link 부분 삭제
위의 url에서 my-sql-pingpong(db 컨테이너명) 대신 우리 도메인 넣어도 서버 db와 연결

Dockerfile

```
FROM openjdk:8-jre-alpine
EXPOSE 8080
ARG JAR_FILE=build/libs/backend-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Jenkins

젠킨스 접속

- http://localhost:9090/로 접속 (위에서 포트를 9090으로 설정.)
- 젠킨스 초기 비밀번호 확인

```
docker logs jenkins
또는
sudo docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

- Plugin 다운로드 GitLab, generic WebhookTrigger, GitLab API, GitLab Authentication, Docker, Docker Commons, Docker Pipeline, Docker API

Backend 프로젝트 생성

- 소스코드 관리: Git
 - Repository URL: <https://lab.ssafy.com/s07-webmobile1-sub2/S07P12A403/>
 - Credentials : gitlab id, password로 추가
- Branches to build: */master

- Build Step: Execute shell

```
docker rm spring -f
cd backend
chmod +x gradlew
./gradlew bootJar
docker build . -t pingpongclass:latest
docker run -d -p 8080:8080 --link mysql-pingpong --name spring pingpongclass:latest
```

- 첫줄인 `docker rm spring -f` 는 첫 실행에서는 작성x (이전에 빌드된 이미지를 삭제하란 의미라 처음에는 삭제할게 없다고 빌드 실패가 뜰 수 있음.)
- `chmod +x gradlew` 빌드할 권한부여
- `./gradlew bootJar` jar 빌드 명령
- `docker build . -t pingpongclass:latest` (중간에 " : " 넣는거 잊지말기) pingpongclass라는 이름으로 도커이미지를 빌드 명령.
- `docker run -d -p 8080:8080 --link mysql-pingpong --name spring pingpongclass:latest` pingpongclass:latest를 8080 포트에서, mysql-pingpong 컨테이너와 연결, 컨테이너 이름은 spring으로 해서 작동 명령
- 주의) mysql-pingpong 컨테이너를 먼저 run 후에 be를 run해야됨

OpenVidu CE on premises

openvidu 공식홈페이지 참고: <https://docs.openvidu.io/en/stable/deployment/ce/on-premises/>

```
# 기본 사용 포트
Port configuration in the server
Open these ports (in section Close ports to avoid external attacks you have an UFW sample to configure a firewall)

22 TCP: to connect using SSH to admin OpenVidu.
80 TCP: if you select Let's Encrypt to generate an SSL certificate this port is used by the generation process.
443 TCP: OpenVidu server and application are published by default in standard https port.
3478 TCP+UDP: used by STUN/TURN server to resolve clients IPs.
40000 - 57000 TCP+UDP: used by Kurento Media Server to establish media connections.
57001 - 65535 TCP+UDP: used by TURN server to establish relayed media connections.
```

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

.env수정(오픈비두 설치 시 기본 정보를 가지고 생성)

- 위치는 `/opt/openvidu/.env`

```
DOMAIN_OR_PUBLIC_IP=i7a403.p.ssafy.io

CERTIFICATE_TYPE=letsencrypt

OPENVIDU_SECRET=pingpong403

LETSencrypt_EMAIL=youremail@youremail.com
```

docker-compose.override.yml

- 위치: /opt/openvidu/docker-compose.override.yml
- 우리가 만든 앱 설정 넣어주기

```
client:
  image: whiterisi/client
  restart: on-failure
  network_mode: host
  environment:
    - SERVER_PORT=3000
    - OPENVIDU_URL=http://localhost:5443
    - OPENVIDU_SECRET=${OPENVIDU_SECRET}
    - CALL_OPENVIDU_CERTTYPE=${CCERTIFICATE TYPE}
  logging:
    options:
      max-size: "${DOCKER_LOGS_MAX_SIZE:-100M}"
```

#docker-compose관련 명령어

```
$ docker-compose down
```

```
$ docker-compose up -d
```

```
$ docker-compose logs -f openvidu-server
```

Nginx

- nginx를 직접설치하지않고 openvidu 설치할때 같이 자동설치된 nginx 를 이용
- 우선 nginx 설정파일 경로를 찾아서 접근

```
sudo su
# nginx.conf 파일 위치 찾기
find / -name nginx.conf
결과로 나온 위치 확인 후 해당 경로로 이동하는데 해당 경로는 nginx.conf 파일이있는 경로이고,
우리가 수정할 default.conf은 위에 나온경로/conf.d 아래있음

#directory이동
cd 경로/conf.d

# default.conf 수정
nano default.conf
```

default.conf

- 기본 설정에 client 부분만 수정, 추가 (bold체 부분)

```
upstream client{
  server i7a403.p.ssafy.io:3000
}

server {
  listen 80;
  listen [::]:80;
  server_name i7a403.p.ssafy.io;

  location / {
    rewrite ^(.*) https://i7a403.a.ssafy.io:443$1 permanent;
  }
}
```

```

}

server{
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name i7a403.p.ssafy.io;

    #ssl config

    #Proxy

    location / {
        proxy_pass http://client;
    }
}

```

```

# conf 파일 수정 후 수정사항을 반영
$ docker container exec <container> nginx -s reload
<>에는 현재 가동중인 nginx 컨테이너명 또는 ID를 작성
확인이 필요하다면 docker ps 로 확인

```

Frontend - 수동배포

.env

```

REACT_APP_OPENVIDU_SERVER_URL=https://i7a403.p.ssafy.io
REACT_APP_OPENVIDU_SERVER_SECRET=pingpongclass403

```

Dockerfile

```

FROM node:16-alpine

WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "run", "start"]

```

수동배포

1. package.json 에 "proxy" : "http://i7a403.p.ssafy.io:8080/be"
2. node_modules>react-script>config>webpackDevServer.config
const disableFirewall ='true';로 수정
3. .env 확인 (오픈비두 접속을 위해서 필요) /dockerfile 확인 (도커라이징을 위해서 필요)
4. Terminal
docker build . -t (내 도커허브 아이디)/(레포지토리아름)예를들어, whiterisi/mod
docker push(내 도커허브 아이디)/(내가 생성한 레포지토리아름)
5. mobaxterm로 서버접속
docker pull (내 도커허브 아이디)/(내가 생성한 레포지토리아름)
6. 서버 돌아가는지 확인
docker ps
목록에 내가 빌드한 컨테이너가 있는지 확인
7. cd
cd /opt/openvidu
docker-compose up -d
8. http://i7a403.p.ssafy.io/ 으로 접속

방화벽

To	Action	From
--	-----	----
22/tcp	ALLOW	Anywhere
80/tcp	ALLOW	Anywhere
443/tcp	ALLOW	Anywhere
3478/tcp	ALLOW	Anywhere
3478/udp	ALLOW	Anywhere
40000:57000/tcp	ALLOW	Anywhere
40000:57000/udp	ALLOW	Anywhere
57001:65535/tcp	ALLOW	Anywhere
57001:65535/udp	ALLOW	Anywhere
-- OpenVidu 기본 포트		
8064	ALLOW	Anywhere
8080	ALLOW	Anywhere
3306/tcp	ALLOW	Anywhere
OpenSSH	ALLOW	Anywhere
Nginx HTTP	ALLOW	Anywhere