pingpong class

포팅메뉴얼

서울 4반 3팀

포팅메뉴얼

기술스택

개발환경

OS

- window 10
- Ubuntu 20.04 LTS

IDE

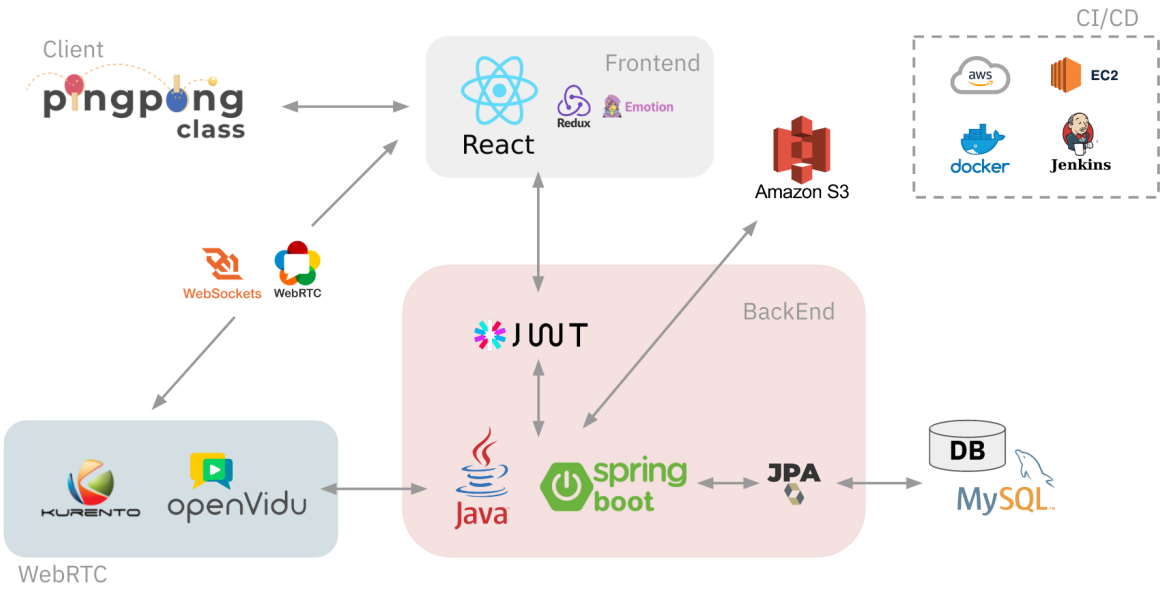
- IntelliJ IDE 2022.1.3
- Visual Studio Code 1.70

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

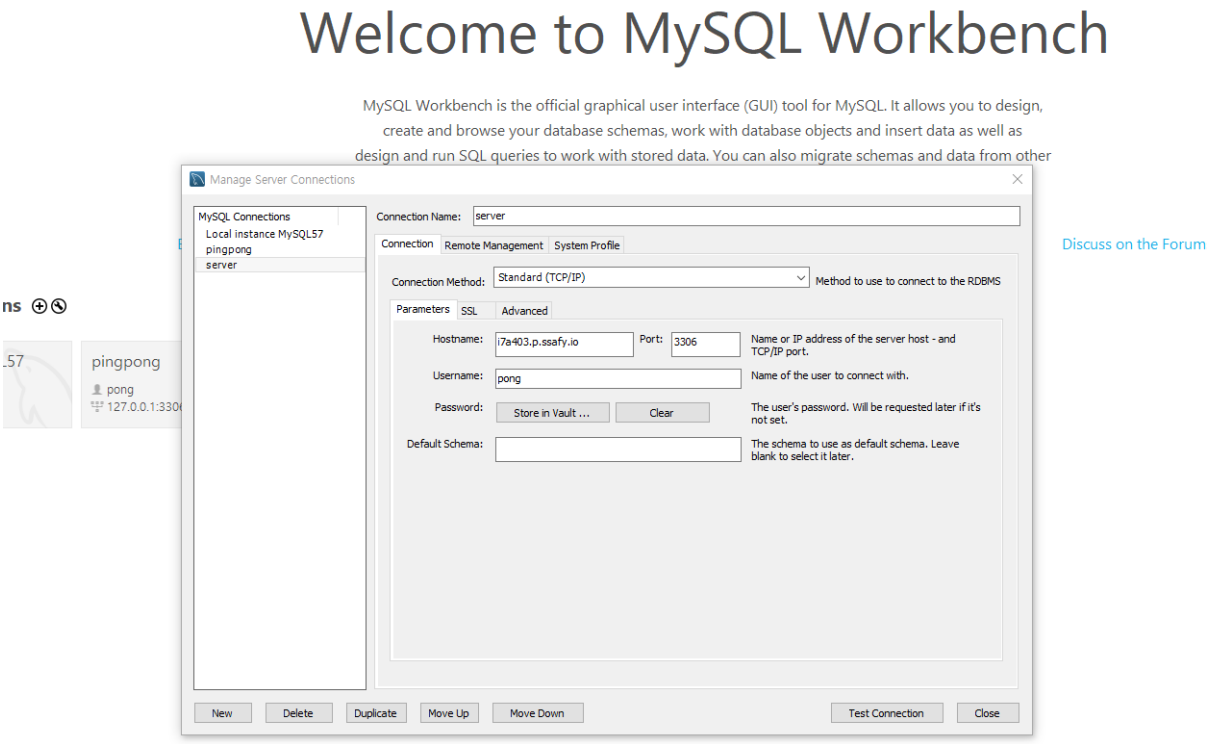
```
docker pull openjdk:8-jre-alpine
```

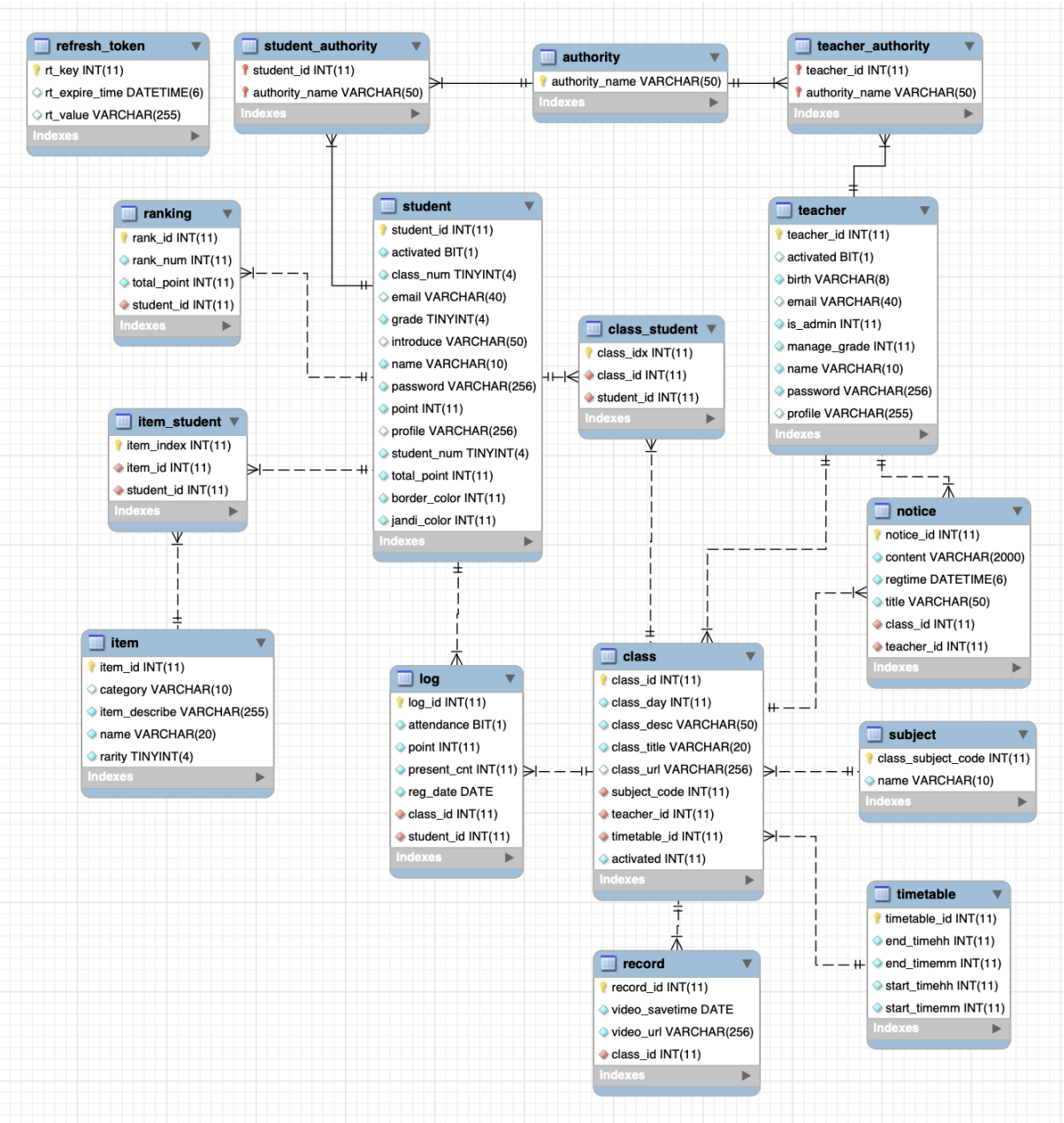
DB

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

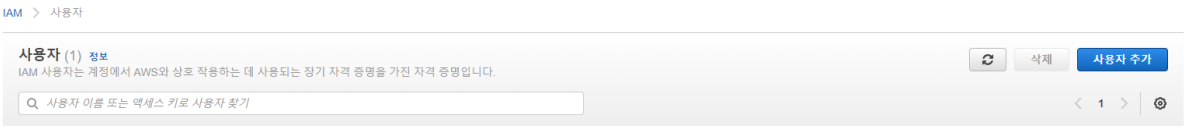




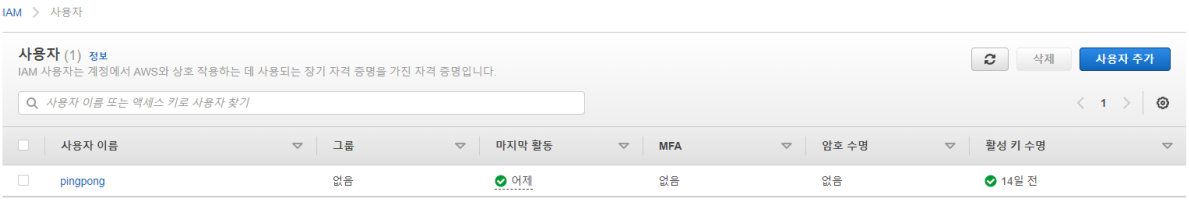
S3

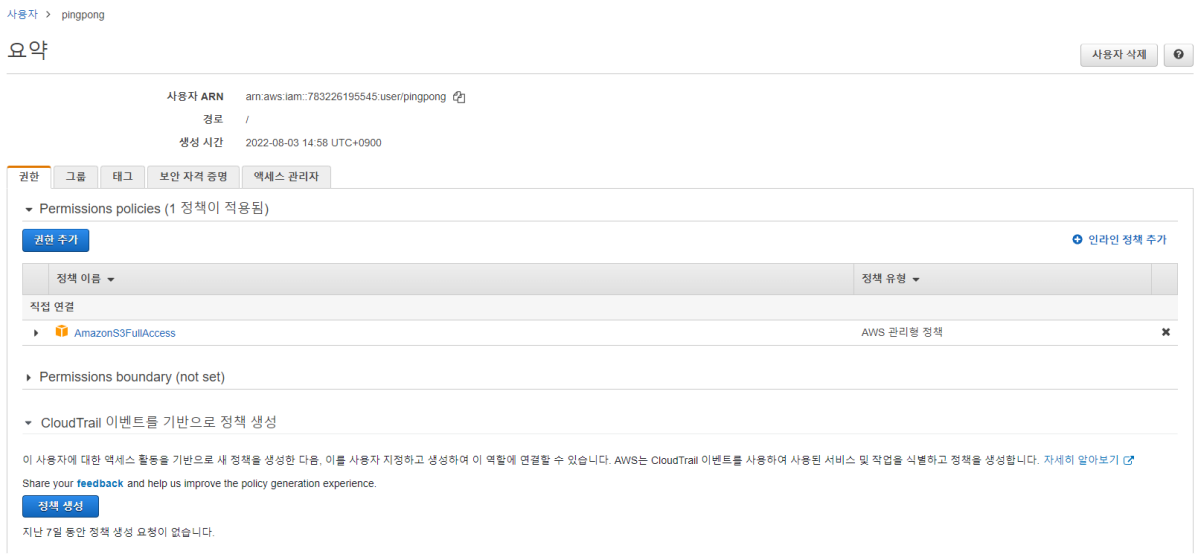
- 아마존 서버 컴퓨터의 저장공간만 빌리는 서비스. 파일 업로드, 다운로드가 가능하다.
- AWS 계정 필요

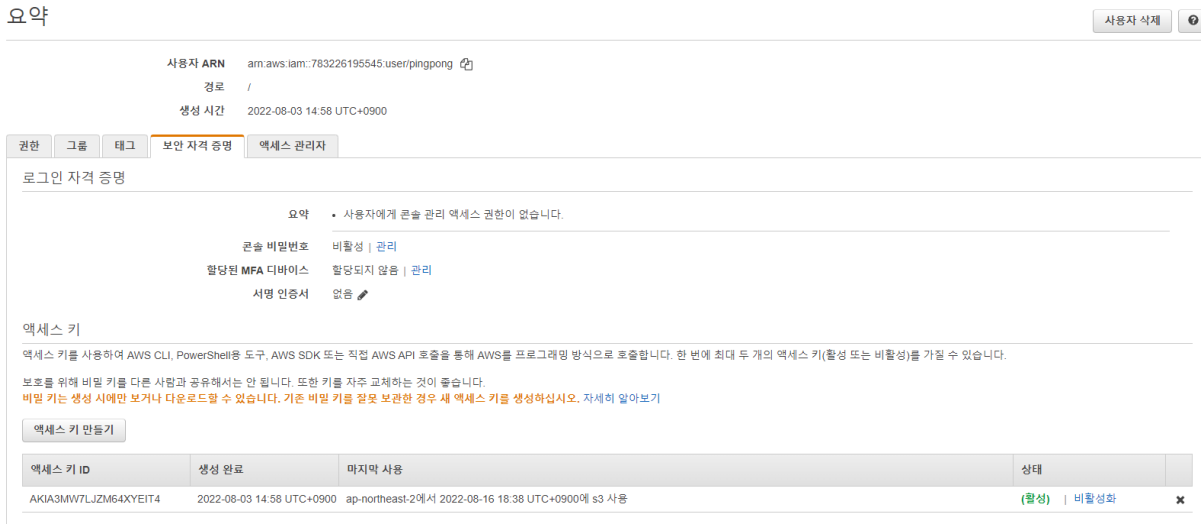
IAM - 사용자

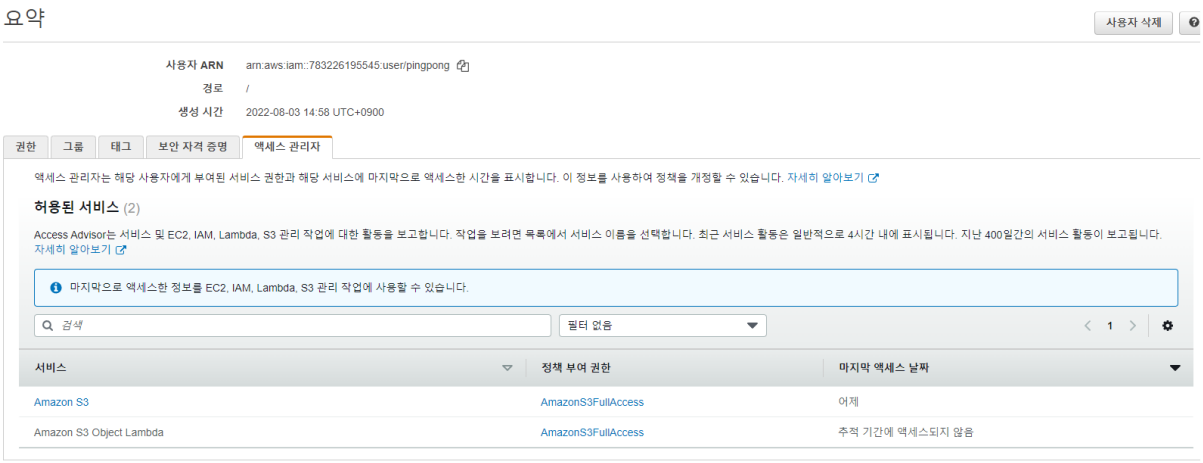


- 사용자 추가 클릭
- 사용자 이름 작성
- AWS 액세스 유형 선택 → 엑세스 키 선택 시 api 로 접근 가능한 key 발급



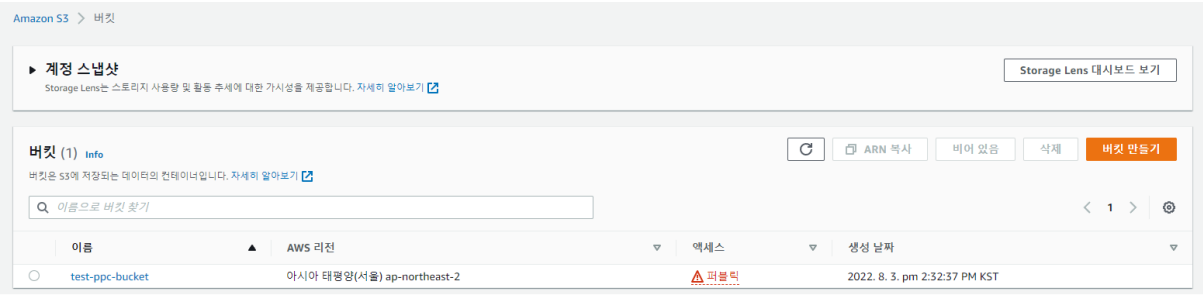


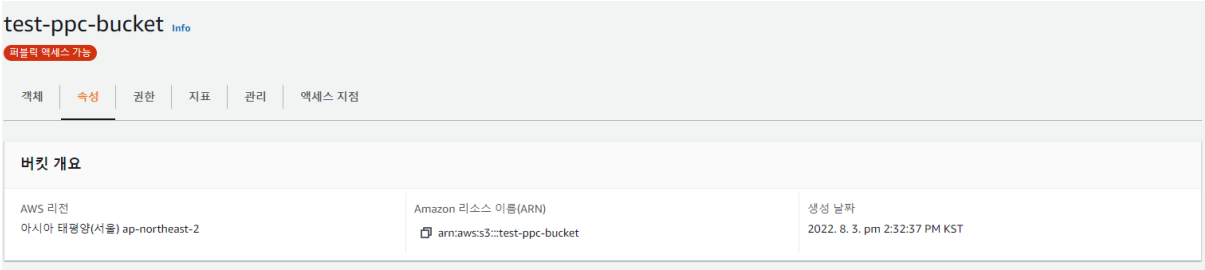




S3 - bucket

- 버킷 만들기





springboot 설정

aws.yml

```
#S3
cloud:
  aws:
    credentials:
      accessKey: AKIA3MW7LJZM64XYEIT4
      secretKey: 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
    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 Comons, 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
- Branchs 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 gadlew 빌드할 권한부여
- ./gradle 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_SECRECT=${OPENVIDU_SECRET}
    -CALL OPENVIDU CERTTYPE=${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 nignx 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)/(레포지토리이름)예를들어, whiterisi/mod
    docker push(내 도커허브 아이디)/(내가 생성한 레포지토리이름)
    5. mobaxterm로 서버접속
    docker pull (내 도커허브 아이디)/(내가 생성한 레포지토리이름)
    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
```