포팅 매뉴얼 (배포 정리)

생성일	@2022년 2월 14일 오후 8:46
작성자	
태그	

1. 기본 환경 설정

1)openjdk version "1.8.0_192"

```
$sudo apt-get update

$sudo apt-get install openjdk-8-jdk

$java -version → java 버전을 확인.
```

2)Mysql 설치 (8.0) → (Ec2 서버내 설치)

```
$sudo install mysql
```

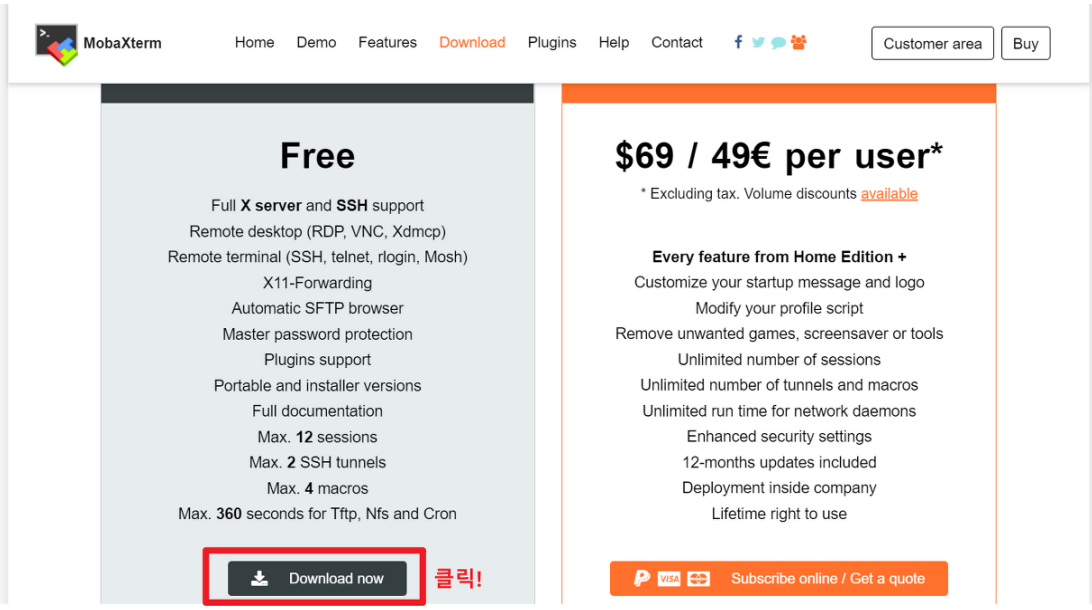
```
$sudo apt-get install mysql-server

$root 계정 및 비밀번호 설정
```

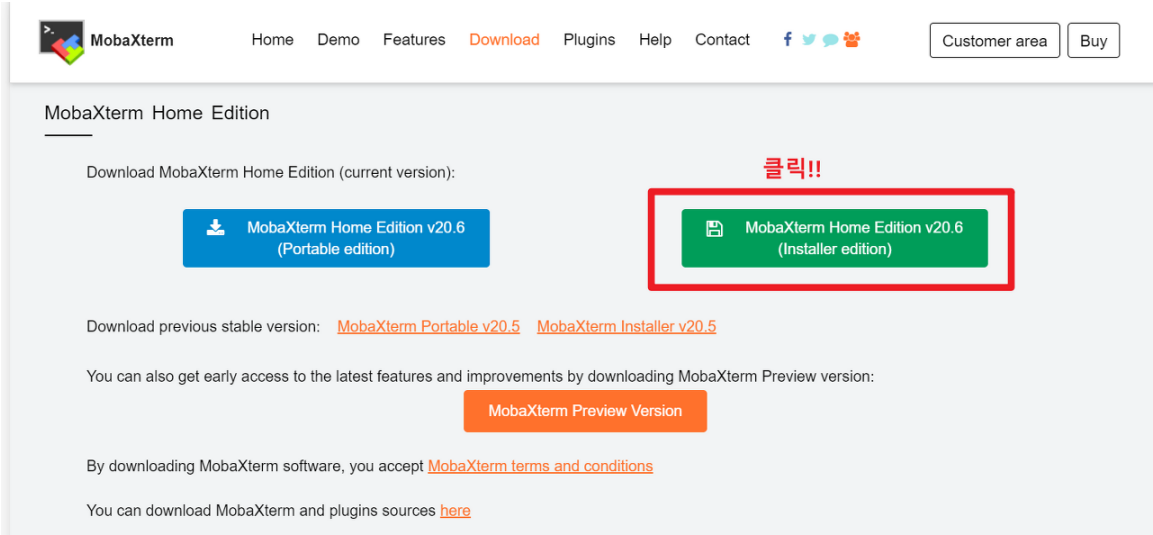
3)Intellij 2021.03 (ultimate/educate) → 명세서를 참고하여 설정 세팅

4)Ec2 서버 사용을 위한 mobaxterm 설치.

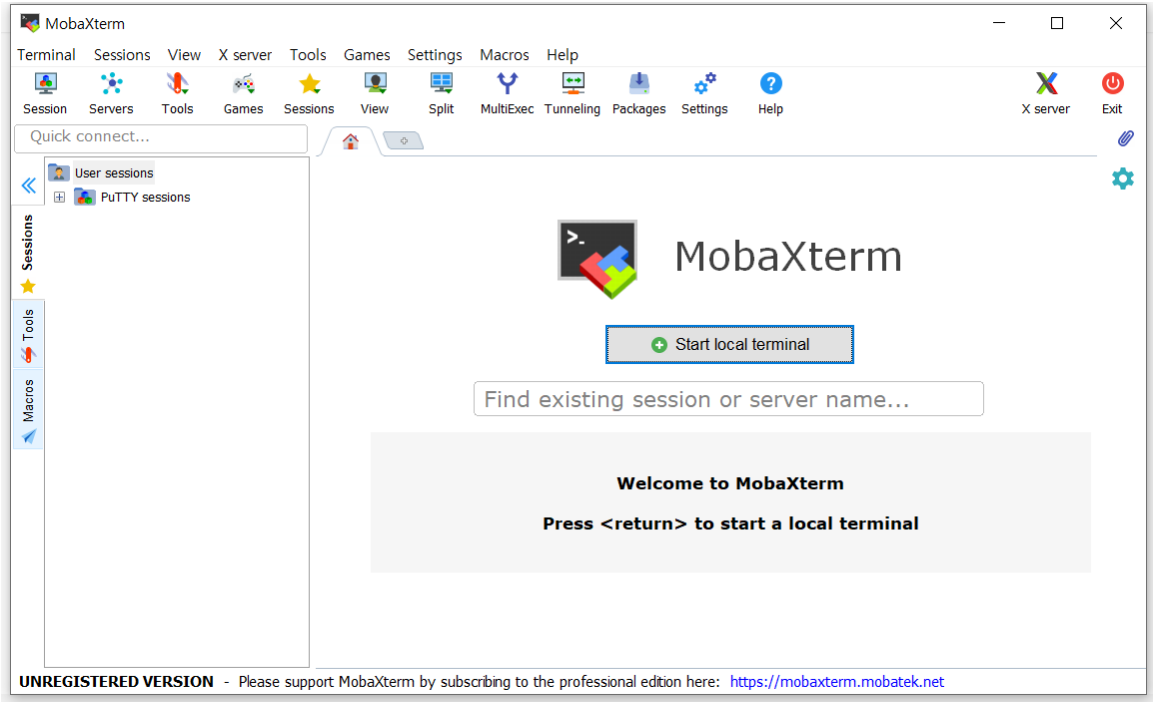
EC2를 세팅하기 위해 mobaXterm을 다운로드 해야 합니다.

mobaXterm : https://mobaxterm.mobatek.net/download.html

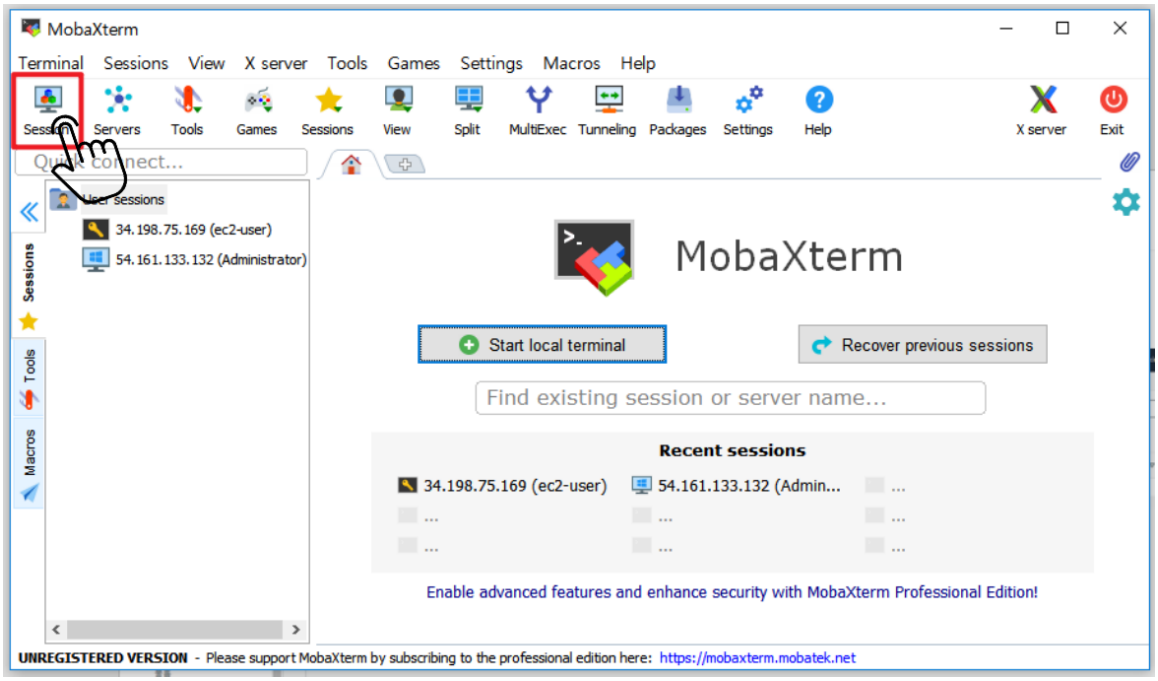




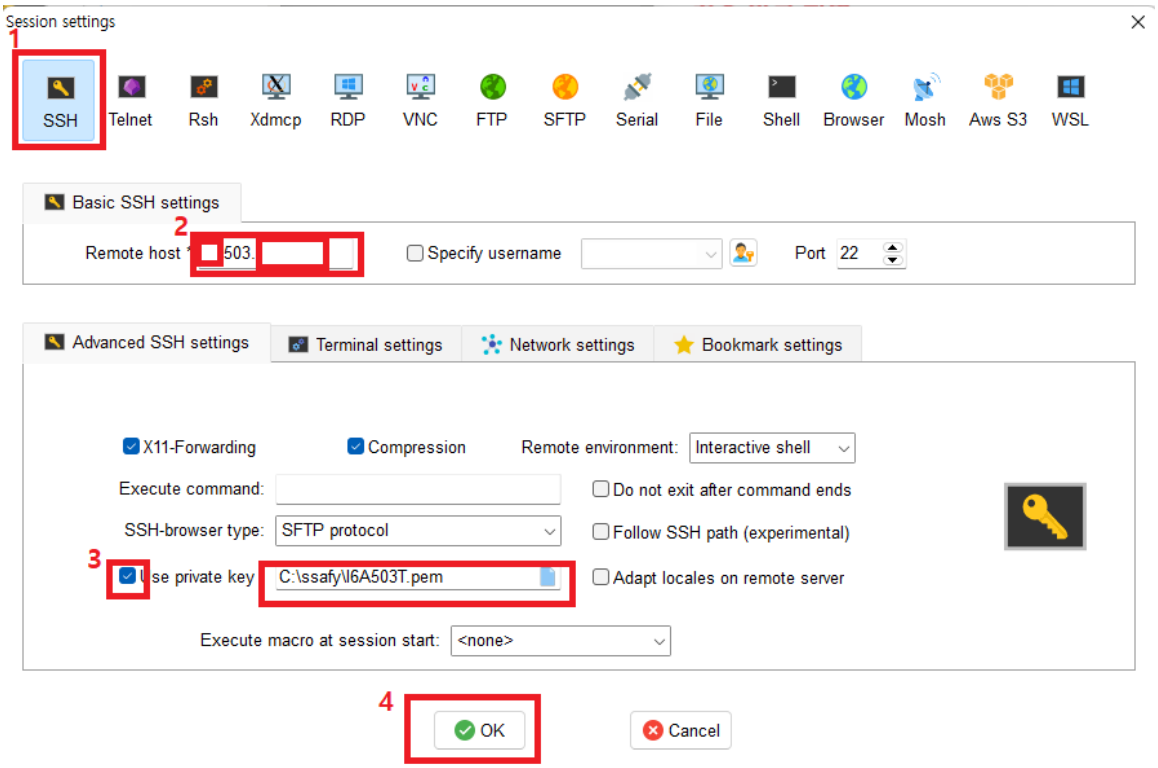
다운로드 받은 파일로 설치(빨간색 박스) 를 하면됩니다.



설치했을때 이런 화면이 나오면 설치 성공!

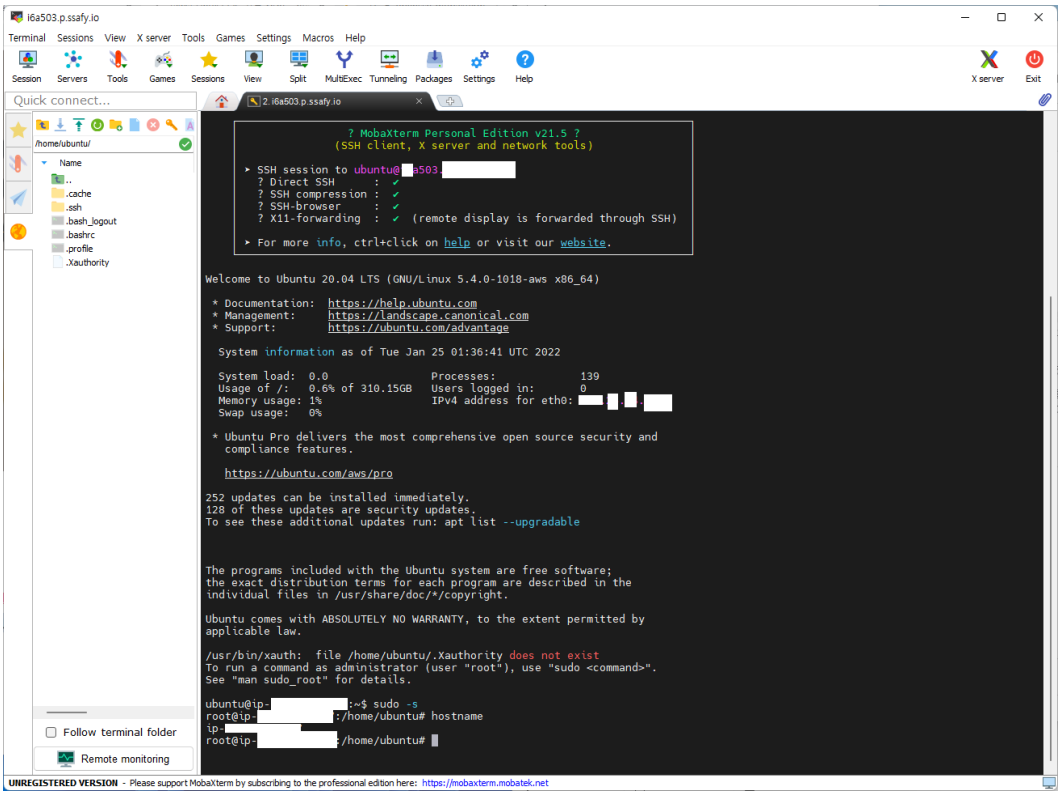


Session 버튼을 클릭



session 화면이 뜨면, 아래 사진을 보고 순서대로 진행

- SSH 클릭
- 접속하고자 하는 인스턴스의 EIP를 입력: ubuntu@i6a503.p.ssafy.io
- 인스턴스에 접속하기 위해서는 PEM파일이 필요. 왼쪽에 체크 후,
  인스턴스를 만들 때 다운로드 받았던 PEM파일 위치를 선택
- OK버튼을 클릭
- login as: ubuntu 입력



이와 같은 화면이 떴으면 접속 완료

2. Nginx 설정/설치 확인

기본환경 - ubnutu 18.01 lts , nginx latest version , ssafy 대여 서버 사용 (i6a503.p.ssafy.io)

nginx 1.18 사용 (ec2 서버에서 설치)

1) 서버 패키지 목록 업데이트

```
$sudo apt update

$sudo apt upgrade

$sudo apt autoremove
```

2) Nginx 다운로드

```
$sudo apt install nginx    // sudo apt remove nginx → 제거
```

3) Ngnix 실행

```
$sudo service nginx start ( stop , restart ,reload 기능 有)

$sudo service nginx status ( nginx 동작 확인 → active 초록불 )
```

4) Nginx 버전 확인

```
$nginx -v ( 보통 엔간한 프로그램 버전확인 *** -v / *** -version 으로 다 되긴 하는듯)

$sudo dpkg -l nginx → 버전 상세 확인
```

5)Nginx 경로 확인!

```
$sudo find / -name nginx.conf   -> 보통은 root/etc/nginx 폴더에 설치가 됩니다!!
```

6)nginx 환경설정 → nginx 경로에 있는 sites-avaiable 폴더의 dafault 를 수정 (버전마다 다름)

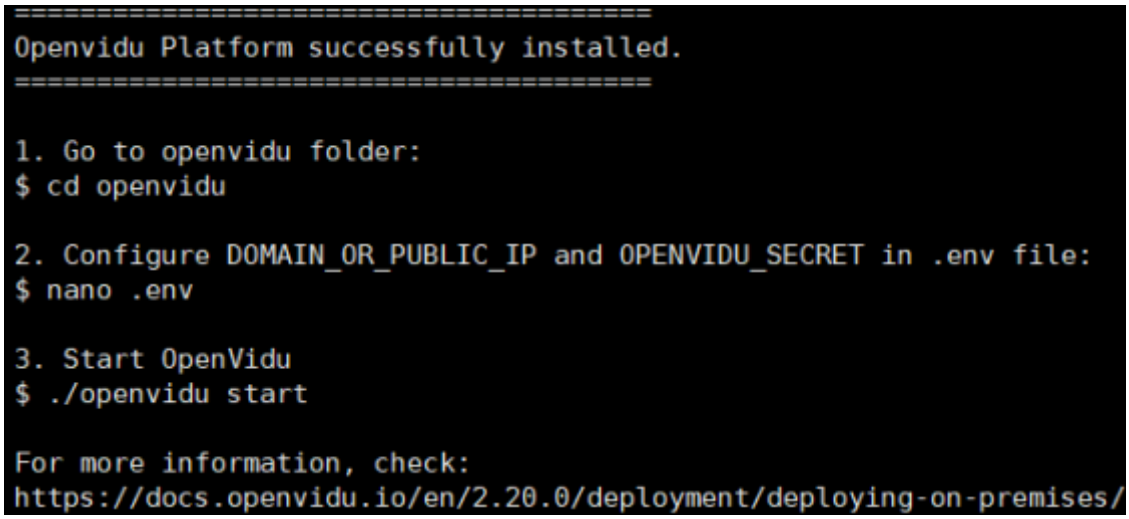
```
vi etc/nginx/sites-available/default

----> default 설정 <-----


    listen 80 default_server;
    listen [::]:80 default_server;

    #root /var/www/html; → 기본 root 페이지 (hello nginx! 같은 페이지가 나옴)
    root /home/ubuntu/dist; → 프로젝트 적용할 dist 폴더로 경로 변경

    index index.html index.htm index.nginx-debian.html; → root 안의 해당 파일을 읽음

    server_name [i6a503.p.ssafy.io](http://i6a503.p.ssafy.io/); → 도메인 주소

    location / {   → 로케이션에서 uri 별 다른 페이지를 띄우게 설정 가능
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ /index.html ;
    }

    location /api { // proxy 설정
    proxy_pass [http://localhost:8080/](http://localhost:8080/);
    proxy_http_version 1.1;
    proxy_set_header Connection "";

    proxy_set_header Host $host;
    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-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Post $server_port;

}

########cert bot 관련 설정 (https) ################

#  https 인증 관련 설정 443 포트를 열고 certbot으로 받은 ssl 인증 키를 복제하여 확인

    listen 443;
    ssl on;
    ssl_certificate /etc/letsencrypt/live/i6a503.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i6a503.p.ssafy.io/privkey.pem;

}
```

7)https 인증 관련 설치 (certbot 사용 ssl 인증)

```
$ sudo snap install --classic certbot

$ sudo certbot --nginx
```

이용약관 동의 → 메일수신 여부 → 인증서 설정 도메인 입력

3. Openvidu 설정/설치 확인



1)설정 파일 수정

```
vi .env

    DOMAIN_OR_PUBLIC_IP=*****[.p.ssafy.io]
    OPENVIDU_SECRET=MY_SECRET # 변경 가능
    CERTIFICATE_TYPE=selfsigned
    #HTTP_PORT=80
    #HTTPS_PORT=443
    [LETSENCRYPT_EMAIL=your@email.com]
```

2)OpenVidu-Server Custom Dockerizing (오픈바이두 서버 도커에 올리기)

- 디렉토리 생성 후 진입 : /home/ubuntu/ 에서 $mkdir OpenVidu → `$cd OpenVidu`

- OpenVidu Git clone : $git clone https://github.com/OpenVidu/openvidu.git'

- 받아 진 openvidu 폴더 진입 : $cd openvidu
- 패키지 클린 : $mvn clean install -U
- openvidu/openvidu-server 폴더 진입 : `$cd openvidu-server`
- openvidu-server 패키지 클린 : `$mvn clean install -U`
- 관리자 권한 부여 : `$sudo su`
- 루트 디렉토리 이동 : cd /
- SSL 인증서 경로 진입 : cd /etc/letsencrypt/live/{SSL 인증받은 도메인}/
  (이때, 도메인은 싸피 도메인 ex. [팀코드.p.ssafy.io](http://xn--k5-xc7ij88fj3d.p.ssafy.io/))
- $openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out 팀코드.p.ssafy.io.p12 --name ssafy -CAfile chain.pem -caname root

  → 이때 설정하는 비밀번호 보관해두기!

  → 팀코드.p.ssafy.io.p12 생성 확인

- openvidu-server 파일에 ssl 인증서 복사, 붙여넣기

  - openvidu-server 의 application.properties 가 있는 경로까지 이동

    $cd openvidu/openvidu-server/src/main/resources/

  - ssl 인증서 복붙 : cp /etc/letsencrypt/live/팀코드.p.ssafy.io/팀코드.p.ssafy.io.p12 ./

    → 싸피 도메인 ssl 인증서(/etc/.../ssafy.io.p12)를 현재 경로(./)에 복사하겠다.

```
* application.properties 수정 : $vi application.properties

    server.address=0.0.0.0
    server.ssl.enabled=true
    server.ssl.key-store=classpath:팀코드.p.ssafy.io.p12
```

```
server.ssl.key-store-password=your-password # 위 10번에서 설정한 비번
server.ssl.key-store-type=PKCS12
server.ssl.key-alias=ssafy
server.servlet.session.cookie.name=OVJSESSIONID
logging.level.root=info
spring.main.allow-bean-definition-overriding=true

SUPPORT_DEPRECATED_API=true
DOTENV_PATH=.
DOMAIN_OR_PUBLIC_IP=[팀코드.p.ssafy.io](http://xn--k5-xc7ij88fj3d.p.ssafy.io/)
OPENVIDU_SECRET=MY_SECRET # openvidu .env 설정과 통일
CERTIFICATE_TYPE=selfsigned
HTTPS_PORT=5443 # 포트번호는 자유

KMS_URIS=["ws://localhost:8888/kurento"]
```

- openvidu 최상위 파일로 이동
- 메이븐 빌드 : `$mvn package -DskipTests`
- openvidu-server 파일로 이동 : `$cd openvidu-server`
- 파일 접근 권한 부여 : `$chmode 777 create_image.sh`
- 도커 이미지 생성 : `$./create_image.sh 2.20.1`

  → 이때 2.20.1 은 기존의 openvidu server가 2.20.0 버전으로 도커 실행중이라 구분을
  위해 버전을 달리해줌
- `$cd opt/openvidu`
- 도커 컴포즈 파일 수정 : `$vi docker-compose.yml`

```
services:

    openvidu-server:
        image: openvidu/openvidu-server:2.20.1
        restart: on-failure
        network_mode: host
        entrypoint: ['/usr/local/bin/entrypoint.sh']
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - ${OPENVIDU_RECORDING_PATH}:${OPENVIDU_RECORDING_PATH}
            - ${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:${OPENVIDU_RECORDING_CUSTOM_LAYOUT}
            - ${OPENVIDU_CDR_PATH}:${OPENVIDU_CDR_PATH}
        env_file:
            - .env
        environment:
            - SERVER_SSL_ENABLED=true
            - SERVER_PORT=5443 # application.properties 설정과 일치
            - KMS_URIS=["ws://localhost:8888/kurento"]
            - COTURN_REDIS_IP=127.0.0.1
            - COTURN_REDIS_PASSWORD=${OPENVIDU_SECRET}
            - COTURN_IP=${COTURN_IP:-auto-ipv4}
```

3)실행 하기

- /opt/openvidu 경로에서 $./openvidu start
  → 이전에 실행을 했다면 'stop' 후 'start' 또는 'restart'
- 도커 확인 : docker ps

- 화면에 나오는 URL 접속

OpenVidu is ready!

- OpenVidu Server URL : [https://팀코드.p.ssafy.io:5443]
- OpenVidu Dashboard : [https://팀코드.p.ssafy.io:5443/dashboard]

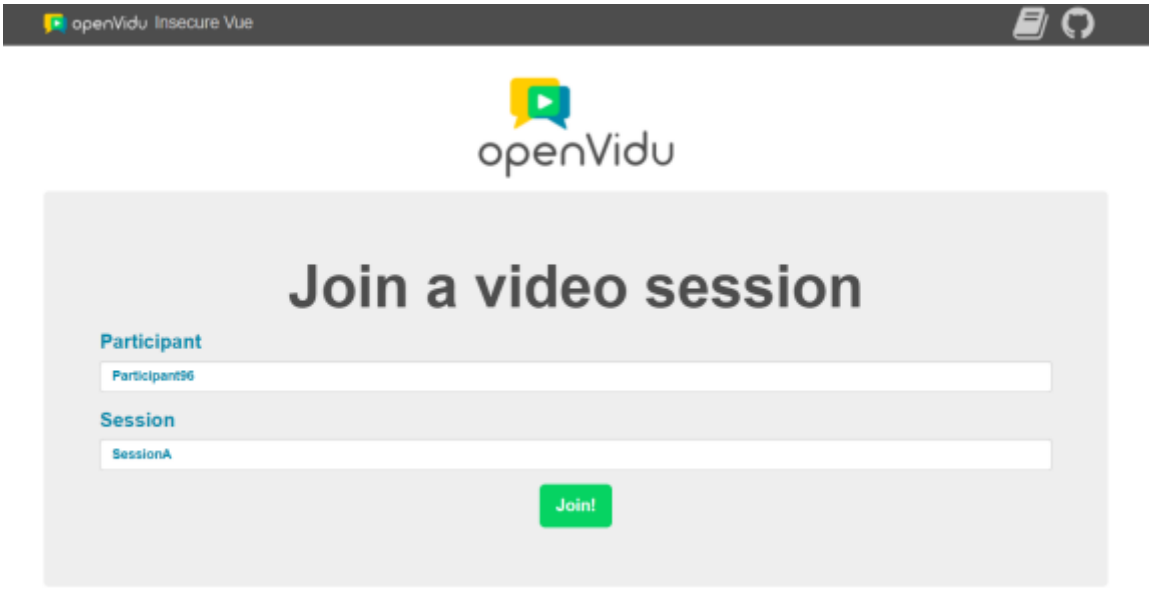
위에 나온 형식으로 사이트 페이지가 떠야합니다

- 화면 확인

- Git 에서 tutorial 클론 :git clone https://github.com/OpenVidu/openvidututorials.git
- openvidu-insecure-vue 파일을 VSCode로 열기
- /src/App.vue 코드 수정

```
const OPENVIDU_SERVER_URL = "https://팀코드.p.ssafy.io:5443";
const OPENVIDU_SERVER_SECRET = "MY_SECRET"; // 본인이 위에서 지정한 비밀번호
```

- npm 설치 후 실행 : npm i → npm run serve
- 맨 처음 실행 시 비밀번호 입력 팝업 : .env 에서 설정한 SECRET 입력
- 화면 확인



- 브라우저에서 카메라/마이크 알림 권한 알림이 뜨면 허용
- 본인의 캠과 마이크가 잘 나오면 성공
- 이후 OpenVidu 공식문서의 튜토리얼을 참고하여 개발

- 최종 파일 구조

  /home/ubuntu/OpenVidu/openvidu : Git clone 받아 openvidu-server dockerizing

  /home/ubuntu/opt/openvidu : curl 로 설치 ./openvidu start 실행

4. git clone 이후 배포전 설정 사항

1) 프론트 (vscode 기준)

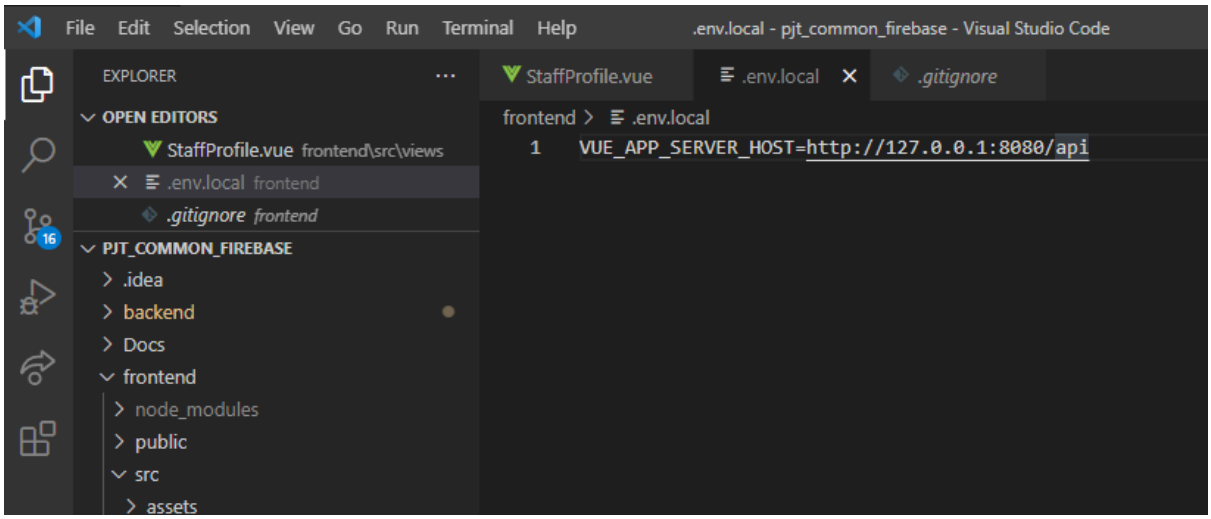
- nodemodules 폴더 삭제 (폴더가 있을 시 삭재 없으면 상관 x)

- /frontend/ 위치에서 npm install 을 입력한다.

```
Hyunkyu@DESKTOP-TV4EOBP MINGW64 ~/develop/ssafy/pjt_common_firebase/frontend (test)
$ npm install
```

- /frontend/.env.local 파일을 생성하여 SERVER URL을 환경파일에 등록해준다.

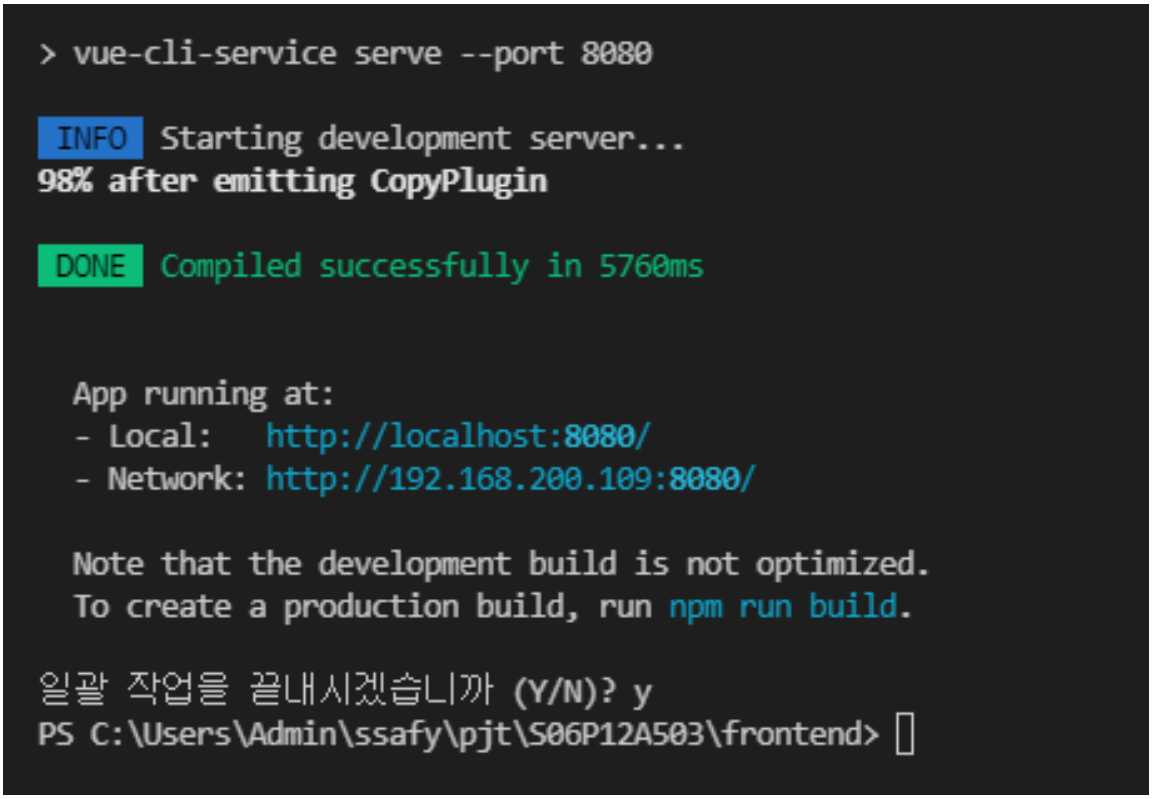
```
`VUE_APP_SERVER_HOST=https://i6a503.p.ssafy.io/api
```



- npm install 실행

위의 방법이 안될경우 다음을 실행 후 재실행

```
* npm install --save @popperjs/core (설정 파일 설치)

* npm run serve
```



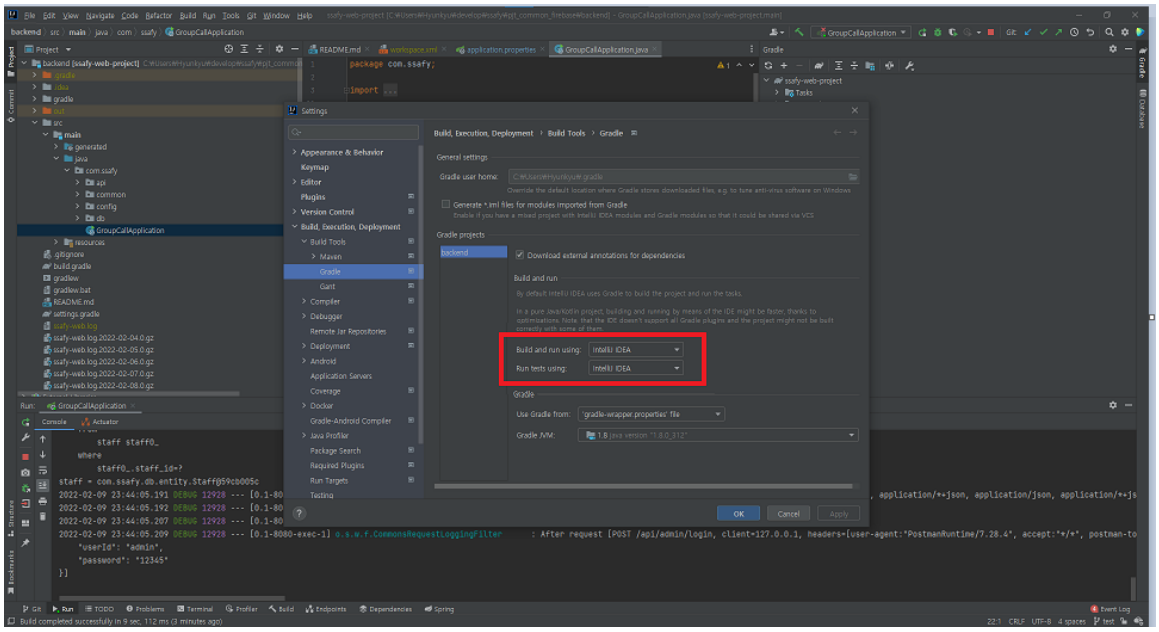

```
* npm run build
```

####(frontend/dist 폴더 → EC2 서버에서 Nginx root에 설정한 곳에 넣음)####

2)백 - (Intelli J 기준)

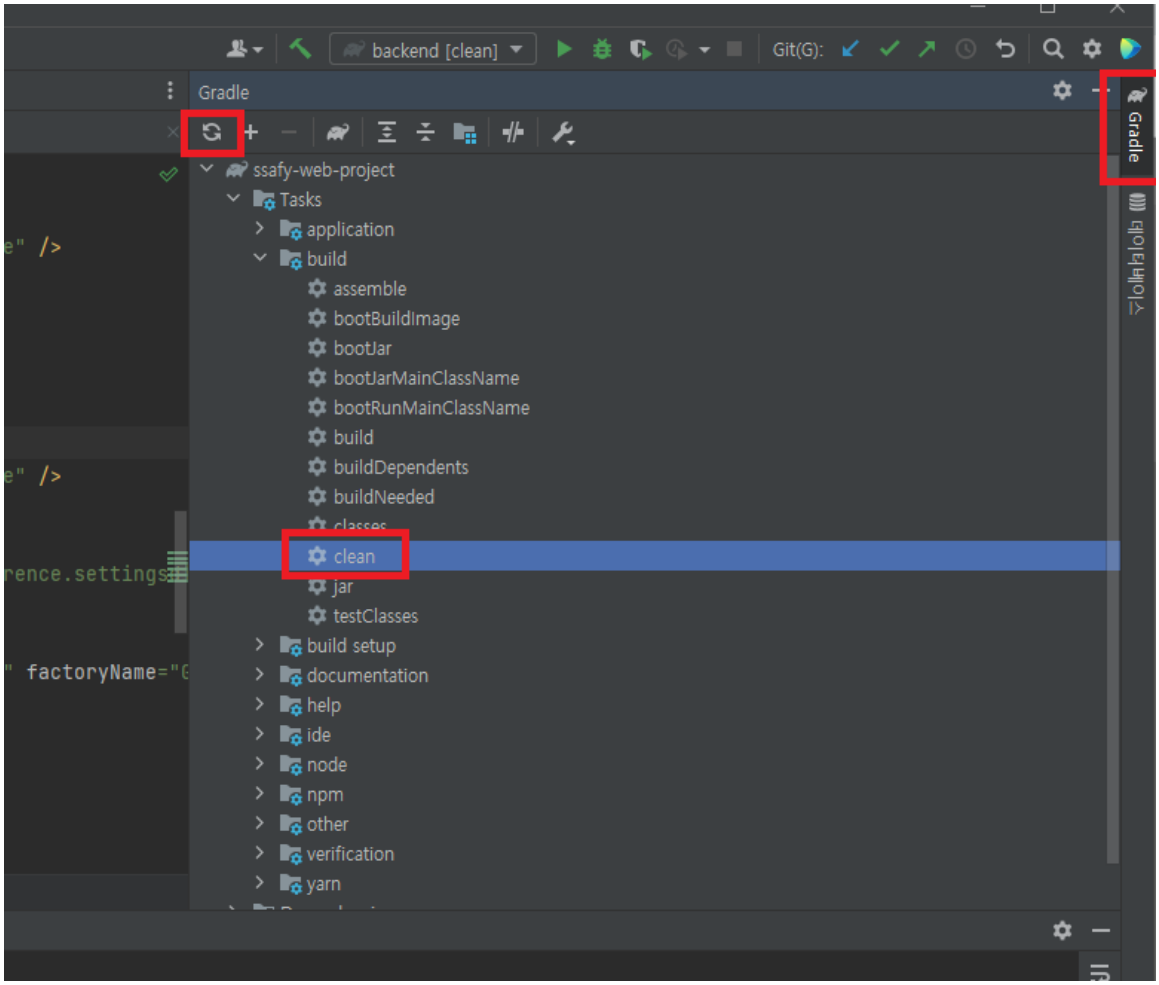
- backend폴더를 IntelliJ에서 열기

- IntelliJ에서 settings에서 gradle 설정을 그림과 같이 IntelliJ IDEA로 변경하고 OK를 누른다.



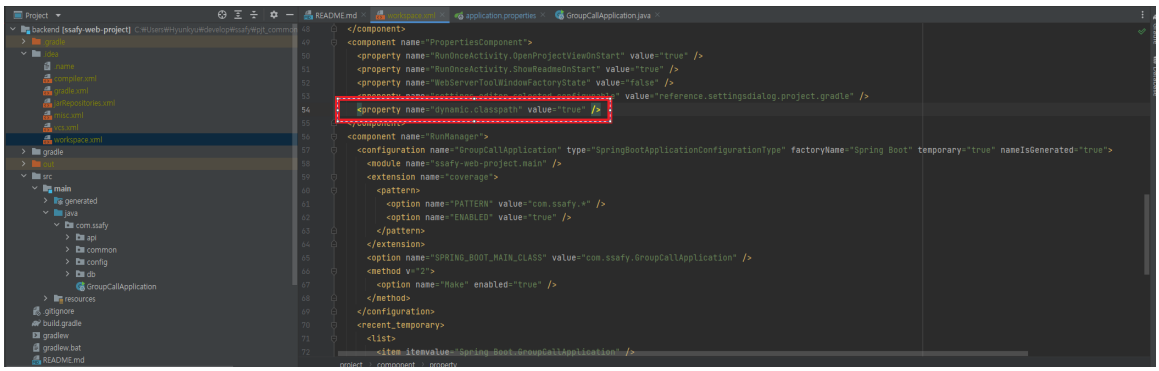
- 우상단 gradle 아이콘 (코끼리) 클릭

  → refresh(재활용 아이콘) 후 build탭→ clean 실행 하고 다시 refresh

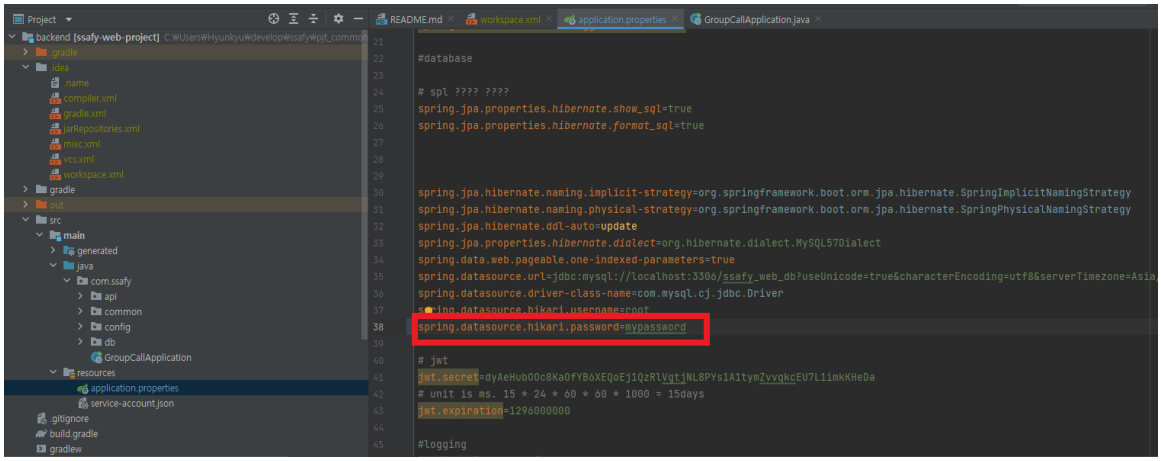


- backend/.idea/workspace.xml에서 `<component name="PropertiesComponent">` 부분에

```
<property name="dynamic.classpath" value="true" /> 추가하고 저장
```

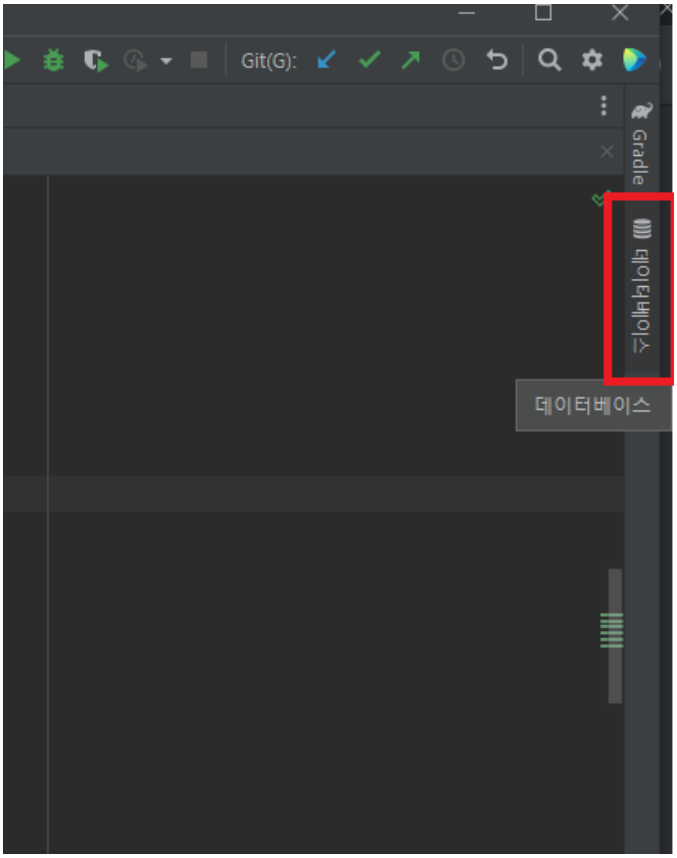


- backend/src/main/resources/application.properties에서 mySQL 설정 공통 DB 변경

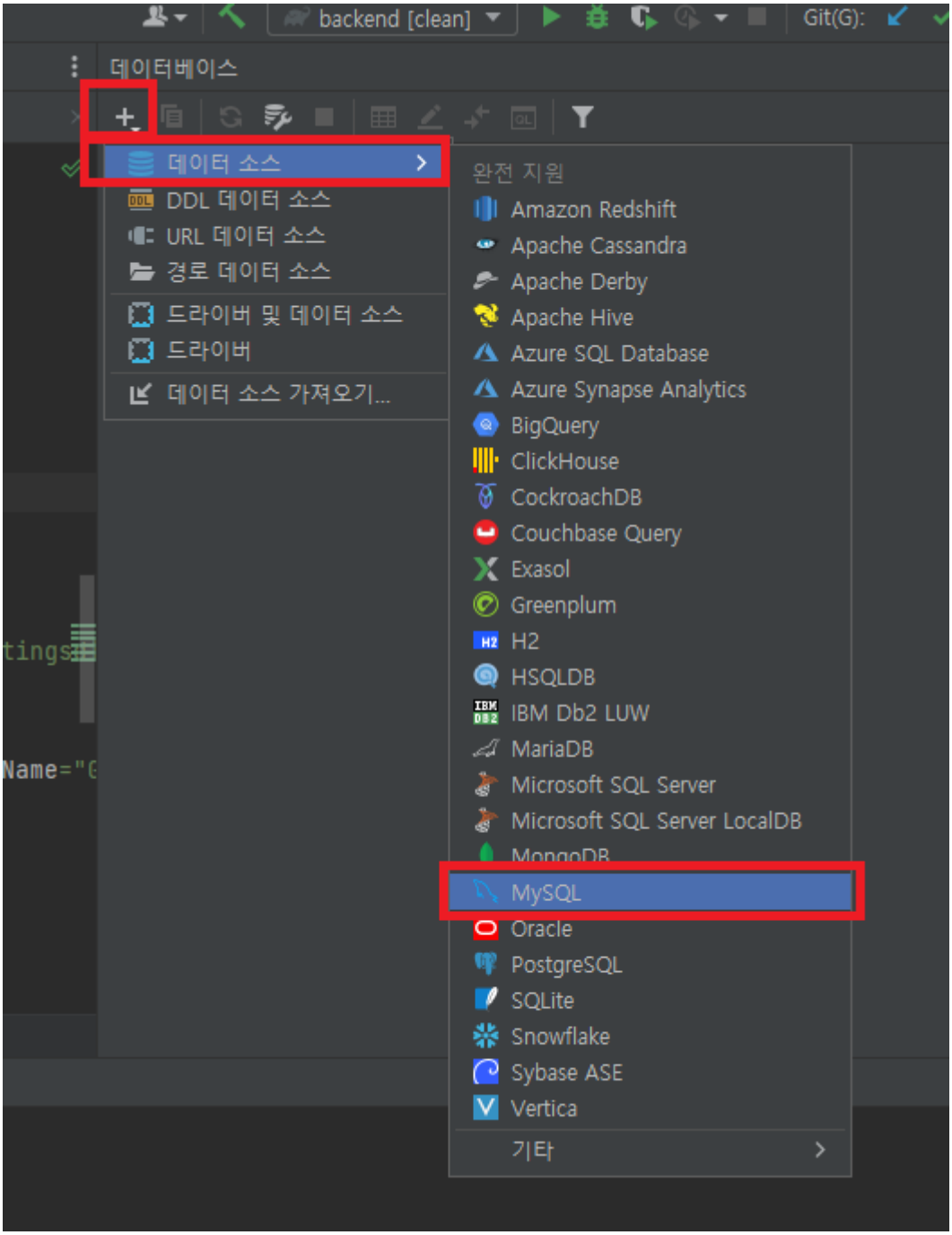


- intelliJ 랑 mysql 연동 (datasource url 가져와야함)

  - 데이터 베이스 클릭



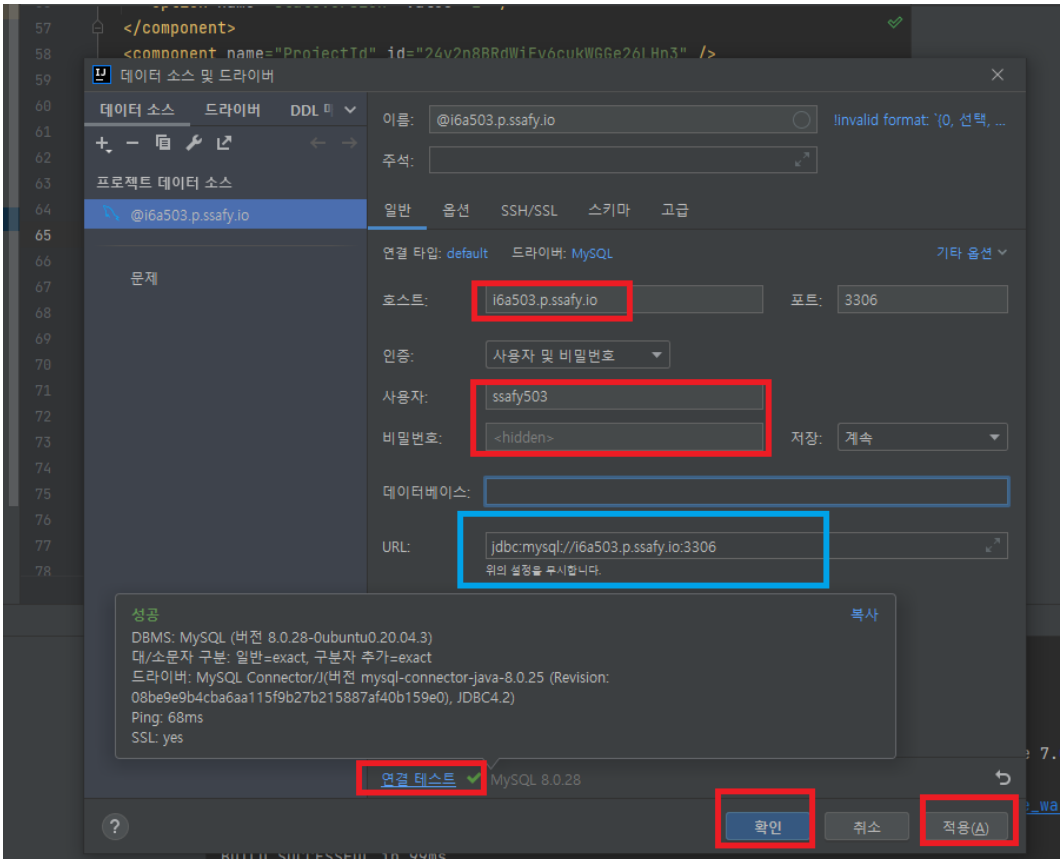
  - 위에서 부터 순서대로 클릭 ('+' -> '데이터 소스' -> 'MySQL')



- 설정 입력 (workbench 연동이랑 비슷)

- datasource url (application.properties 에 넣기) → jdbc:mysql://i6a503.p.ssafy.io:3306

- 호스트 입력 (도메인) → 사용자 아이디 → 비밀번호 → 연결 테스트 → 적용 / 확인

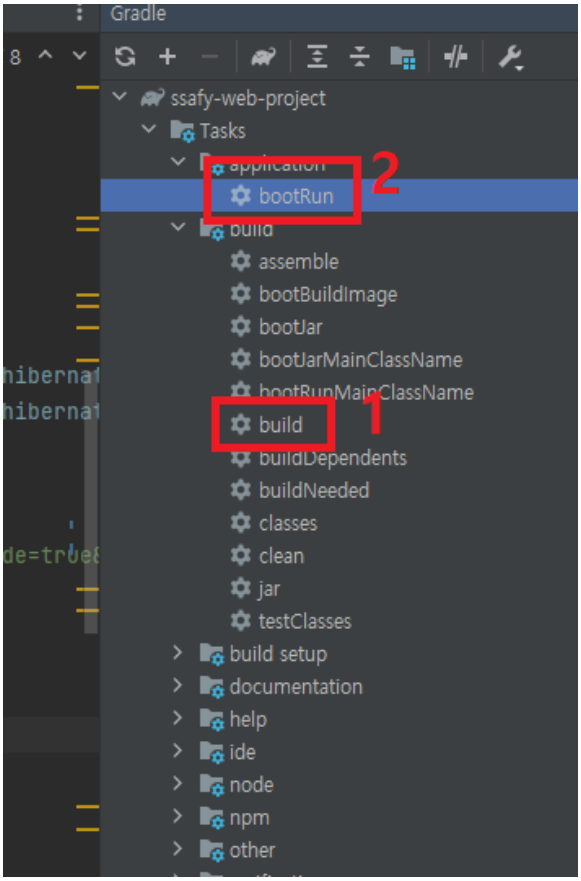


- 빌드 테스트 및 백 서버 실행

  - bootrun (2) 실행 하여 서버가 정상작동 하는지 확인

    → 서버 정상 작동을 확인하고 build 실행하여 jar 파일 생성

    backend/build/libs/****.jar 파일



- 프론트/백 실행 되는거 확인한 후 빌드 파일을 배포!

5. 배포 유의 사항

sudo 사용 자제... 권한 관련 에러 (ex : permission denied) 같은거뜰때만 sudo
mobaxterm 을 사용해 Ec2 서버 계정 접속 후 진행
mobaxterm은 파일 드래그 & 드롭이 가능

1) 배포 작업시에는 nginx 종료 (service nginx stop)

- 배포 끝나고 다시 실행

2) 프론트 엔드 배포 (frontend/dist)

- 프론트에서 빌드 후 dist 파일 백업 하기 → 백업 파일을 정리해서 저장하기!!

- 백업 후 배포할 dist 파일 ec2 서버에 업로드

3) 백엔드 배포 (backend/libs/***.jar)

- 인텔리제이 gradle → task → build 실행

- backend 폴더 내 libs / *******.jar 파일 생성확인 → ec2 서버 키기

- 실행중인 jar 파일 확인 $ps -ef | grep jar -> 있을 시 종료 (kill -9 PID)

- 백업 후 배포할 jar 파일을 업로드 하기 (home/ubuntu)

- jar 파일 ec2서버에서 백그라운드로 실행 $sudo nohup java -jar *****.jar &
  -> 실시간 백엔드 서버 로그 확인을 원할 시 java -jar ****.jar
  -> 백그라운드에서 실행 후 저장된 로그 확인 시 cat nohup.out
- $ps -ef | grep jar 백엔드 서버 실행 확인

4) 배포 완료

- nginx 실행

```
$sudo service nginx start

$sudo service nginx reload (필수 x)

$ps -ef | grep nginx  (nginx 실행 확인)
```