Porting Manual

Index

---

1. Stacks

1.1. Issue Management

1.2. SCM

1.3. Communities

1.4. Development Environment

1. **OS**

2. **IDE**

- IntelliJ IDEA 2022.1.4 (Ultimate Edition)

- Visual Studio Code 1.70.1

3. **Database**

- MariaDB 10.3.34

- MySQL WorkBench 8.0

4. **Server**

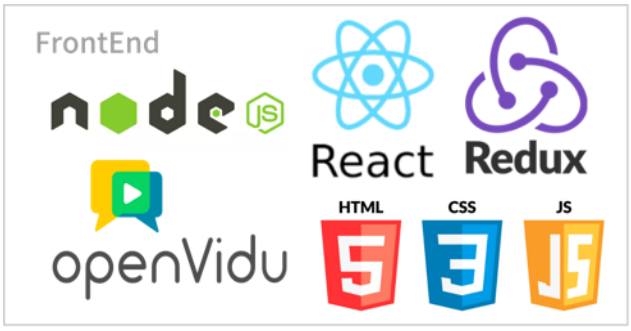
Ubuntu 20.04 LTS

1.5. Details

1.5.1. Back-End

1. Java (Zulu 8.33.0.1-win64)
2. Spring Boot Gradle 7.5
3. Lombok 1.18.24
4. Swagger 3.0.0
5. JPA
6. JWT

1.5.2. Front-End



1. HTML5, CSS3, JAVASCRIPT(ES6)

2. React 17.0.2
3. React-redux 8.0.2, redux-toolkit 1.8.3

1. Node JS 16.16.0
2. OpenVidu 2.22.0

1.5.3. etc

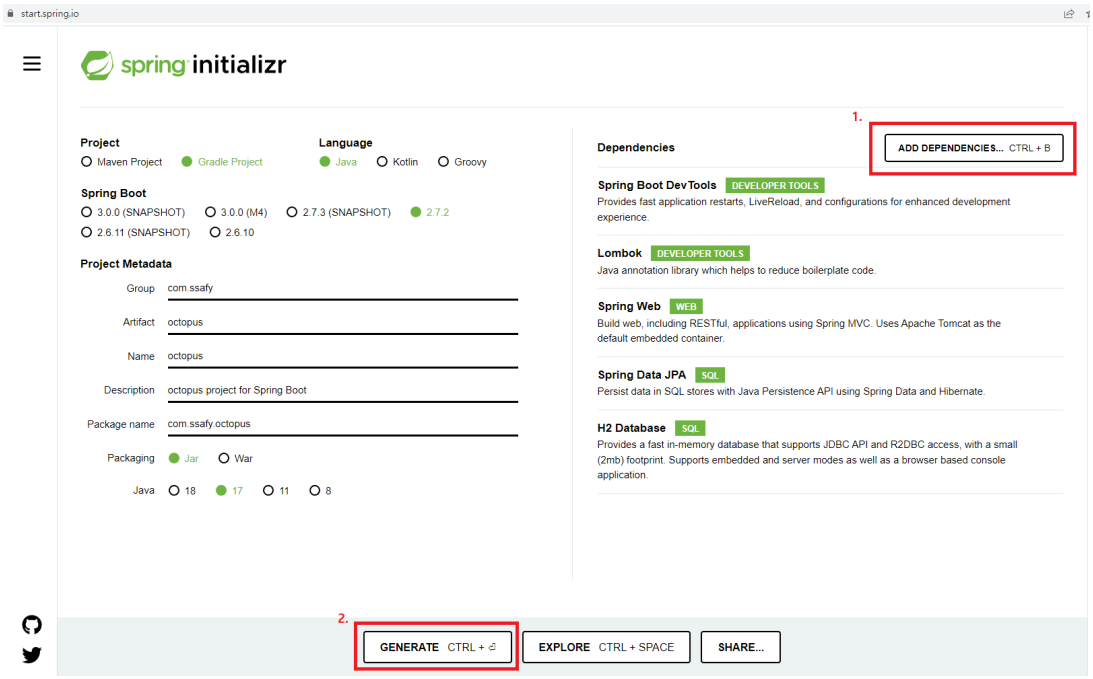
1. AWS EC2
2. Docker (20.10.17)
3. Nginx (1.18.0)
4. certBot

2. Builds

2.1. How to build BE (Spring boot)

2.1.1. GUI

1. **Spring Boot Project Import**
    a. **start.spring.io 사용**
        i. **start.spring.io** 접속하여 원하는 프로젝트를 구성



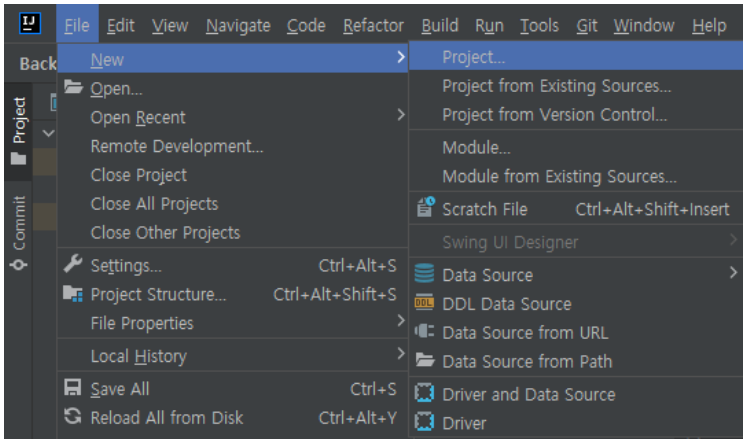
1. **ADD DEPENDENCIES** 를 통해 원하는 **라이브러리 추가**
2. 선택 후 GENERATE 실행하여 zip 파일 다운로드

ii. 다운로드 받은 zip파일 원하는 위치에 압축 해제

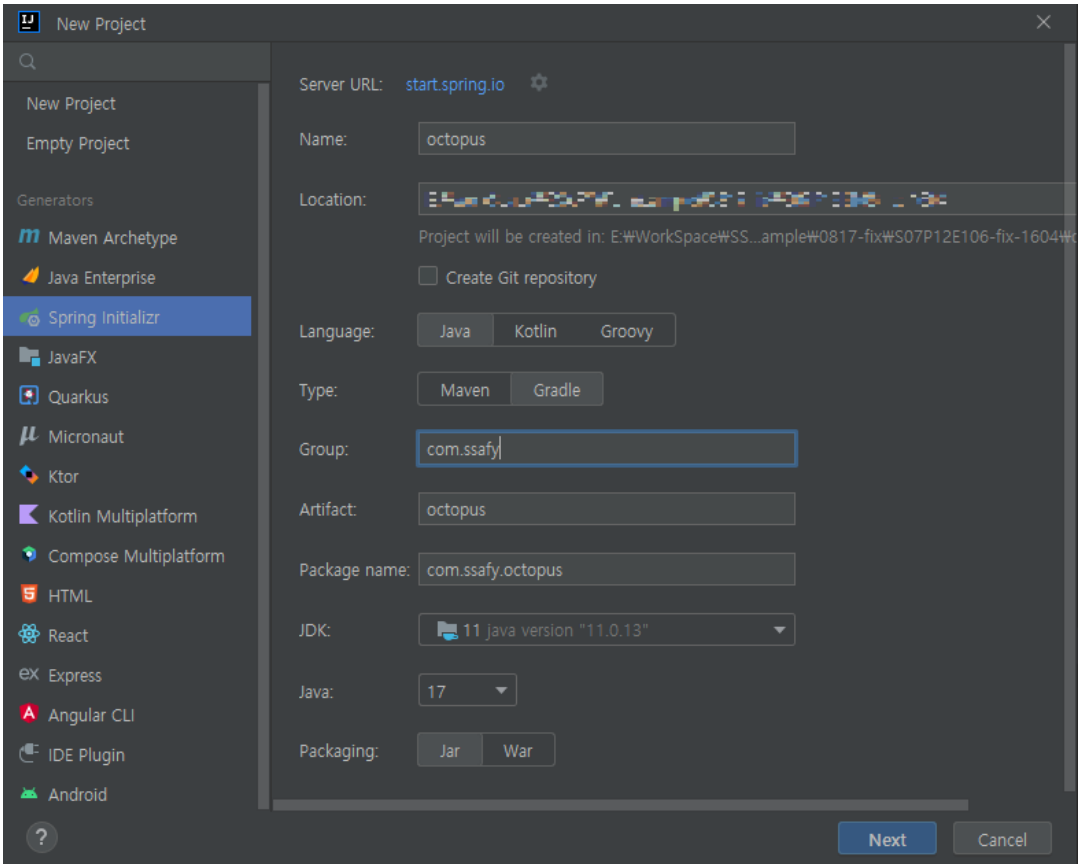
iii. IntelliJ 실행 후 해당 폴더를 Open (혹은 Import)

b. **IntelliJ Ultimate 사용**

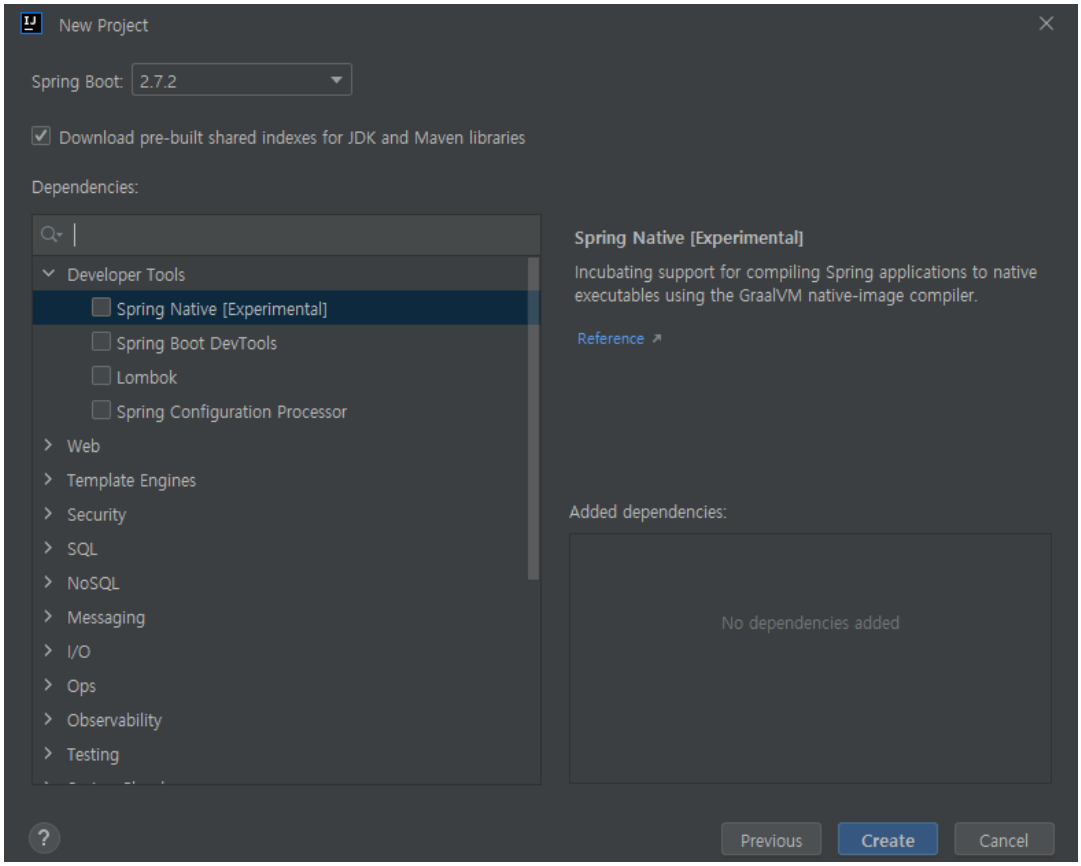
i. IntelliJ 실행 후 File - New - Project 클릭



ii. New Project 에서 원하는 설정으로 세팅 이후 Next

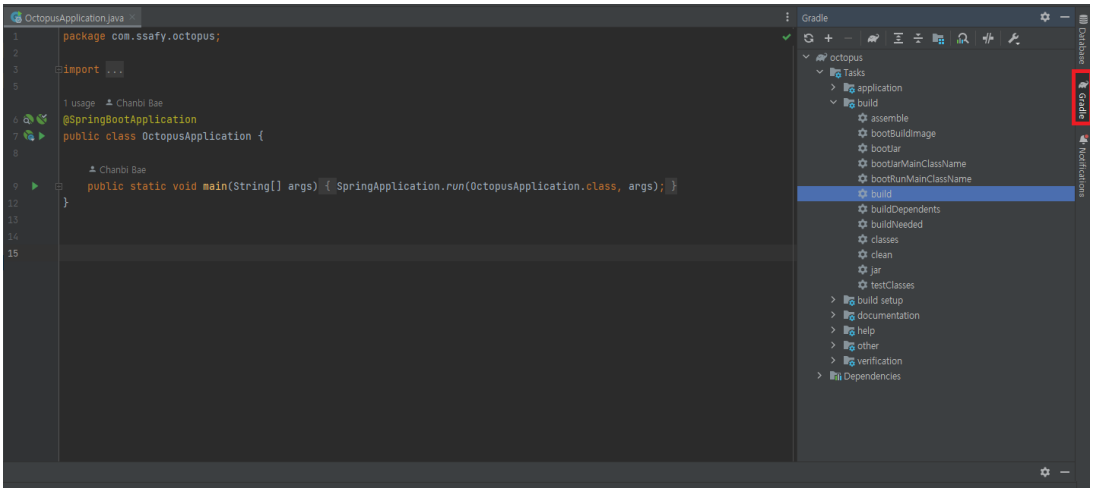


iii. 원하는 라이브러리 추가 이후 Create로 프로젝트 생성

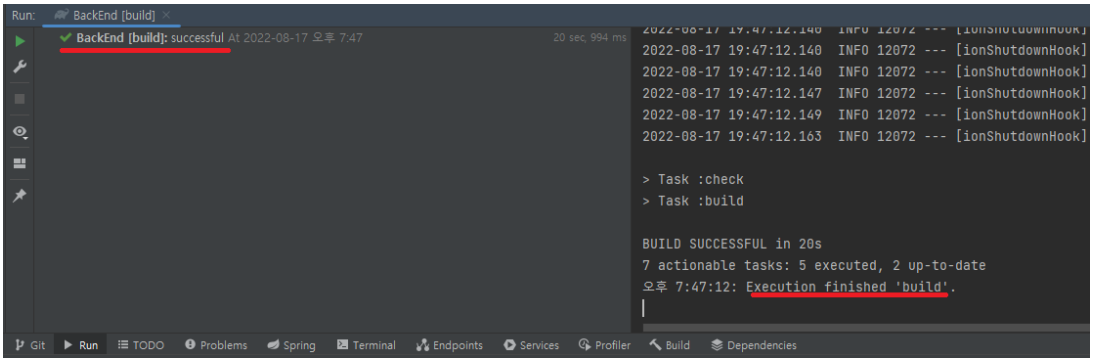


c. **Project Build**

i. 우측 Gradle 클릭 후 Tasks-build-build 클릭



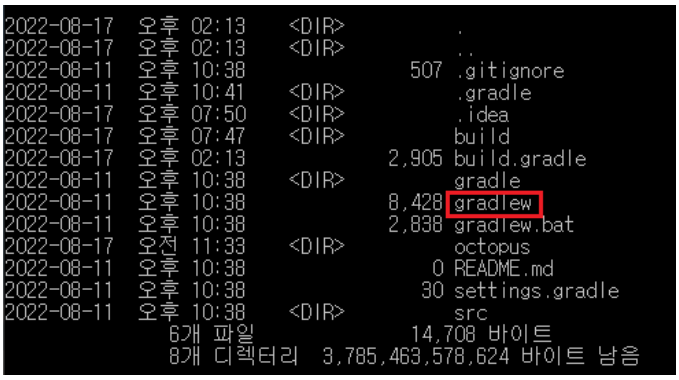
ii. 좌측 하단에 실행 내역(Run)에서 Successful 확인



iii. 프로젝트 폴더 - build - lib - [Project Name]-0.0.1-SNAPSHOT.jar 파일 확인

2.1.2. Command

1. 빌드 희망하는 프로젝트 폴더에서 **gradlew 파일이 존재하는 위치**로 이동



일반적으로 React 폴더 최상단에 존재

2. 아래의 명령어를 통해서 빌드
   a. Windows

```
$ gradlew build
```

   b. Linux

```
$ ./gradlew build
```

2.2. How to build FE (React)

※ 사전에 Node JS가 설치 되어 있어야 함.

2.1.1. React

1. React를 설치 하고자 하는 폴더로 이동

2. cmd 창에 아래 명령어를 실행

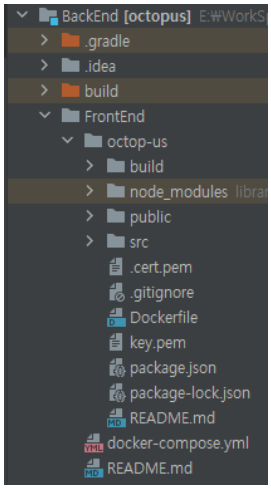
```
npx create-react-app [project name] //project name 이름으로 하위 폴더 생성 후 React 생성
```

```
npm run build // React 빌드
```

2.1.2. Integrated build (BE + FE)

※ Spring Boot, React Project가 다 생성되어 있다는 가정하고 진행.

1. Spring Boot 폴더 아래 React 폴더 생성 후 React 프로젝트 복사(혹은 이동)



BackEnd(Spring Boot 폴더 명) 하위에 FrontEnd 폴더 복사

2. **package.json 파일에서 프록시 설정 (백엔드 포트 번호로 작성)**

package.json 괄호 안에서 제일 밑에 추가

3. Spring Boot의 build.gradle 에 아래 내용 추가

```
// 경로 맞춰서 설정해주기 ($projectDir/FrontEnd/octop-us 이 부분)

def frontendDir = "$projectDir/FrontEnd/octop-us"

sourceSets {
    main {
        resources {
            srcDirs = ["$projectDir/src/main/resources"]
        }
    }
}

processResources {
    dependsOn "copyReactBuildFiles"
}
```

```
task installReact(type: Exec) {
    workingDir "$frontendDir"
    inputs.dir "$frontendDir"
    group = BasePlugin.BUILD_GROUP
    if (System.getProperty('os.name').toLowerCase(Locale.ROOT).contains('windows')) {
        commandLine "npm.cmd", "audit", "fix"
        commandLine 'npm.cmd', 'install'
    } else {
        commandLine "npm", "audit", "fix"
        commandLine 'npm', 'install'
    }
}

task buildReact(type: Exec) {
    dependsOn "installReact"
    workingDir "$frontendDir"
    inputs.dir "$frontendDir"
    group = BasePlugin.BUILD_GROUP
    if (System.getProperty('os.name').toLowerCase(Locale.ROOT).contains('windows')) {
        commandLine "npm.cmd", "run-script", "build"
    } else {
        commandLine "npm", "run-script", "build"
    }
}

task copyReactBuildFiles(type: Copy) {
    dependsOn "buildReact"
    from "$frontendDir/build"
    into "$buildDir/resources/main/static"
}
```

4. 빌드 및 결과물(Project Name - build - lib - jar파일) 확인

3. Deployment Command

3.1. OpenVidu Server

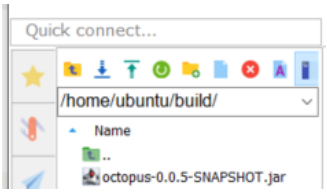
1. OpenVidu 설치된 폴더 이동 (/opt/openvidu)

2. 아래 명령어로 실행

```
./openvidu start
```

3.2. Front & Back End Server

1. 빌드 파일 위치한 폴더 이동 (/home/ubuntu/build)



2. 아래 명령어로 실행

```
java -jar [Server File Name].jar
```

3.3. Nginx Web Server

1. 상태 확인

```
sudo systemctl status nginx
```

2. 프로세스 시작

```
sudo systemctl start nginx
```

3. 프로세스 종료

```
sudo systemctl stop nginx
```

4. 프로세스 재시작

```
sudo systemctl restart nginx
```

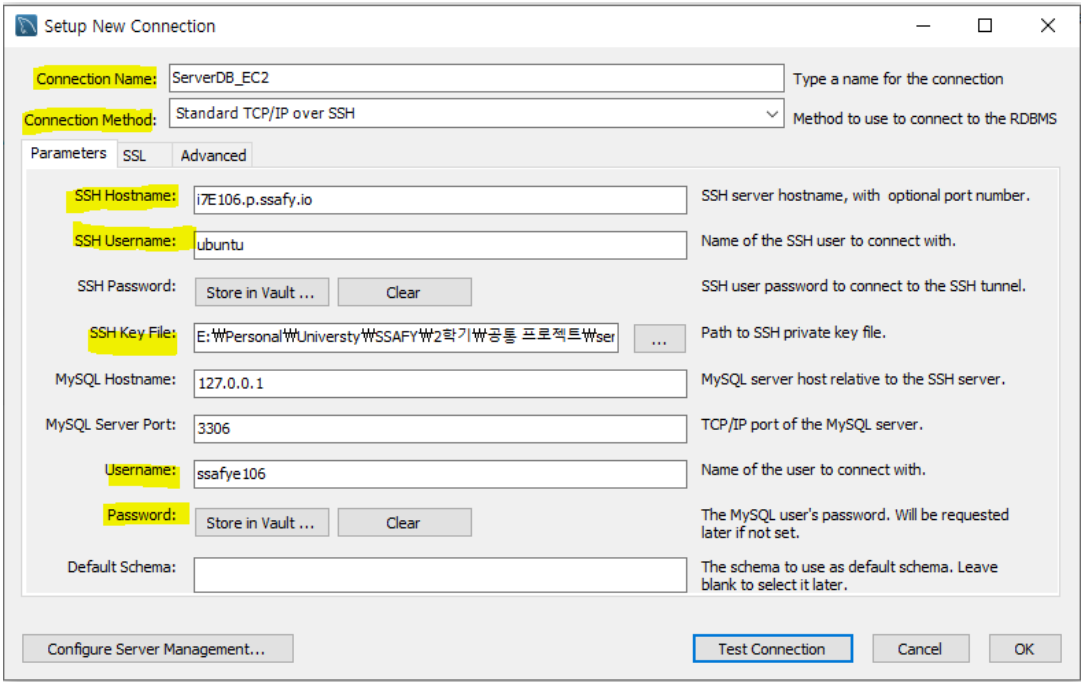
4. How to use the MySQL workbench

4.1. Standard TCP/IP over SSH Connection

1. MySQL Workbench 연결

2. 홈 화면에서 MySQL Connections 에서 + 추가

3. 아래 사진처럼 설정.



- Connection Name : 본인이 해당 커넥션이 어떤건지 알아보기 쉽게 설정
- Connection Method : Standard TCP/IP → Standard TCP/IP over SSH 로 변경
- SSH Hostname : DB 서버 도메인 명으로 설정
- SSH Username : ubuntu 로 설정
- SSH Key File : 공유 받은 pem 파일로 설정 (SSH Password는 설정 안함.)
- Username : 계정 ID 으로 설정
- Password : 계정 PW 로 설정

4. Test Connection으로 연결 확인

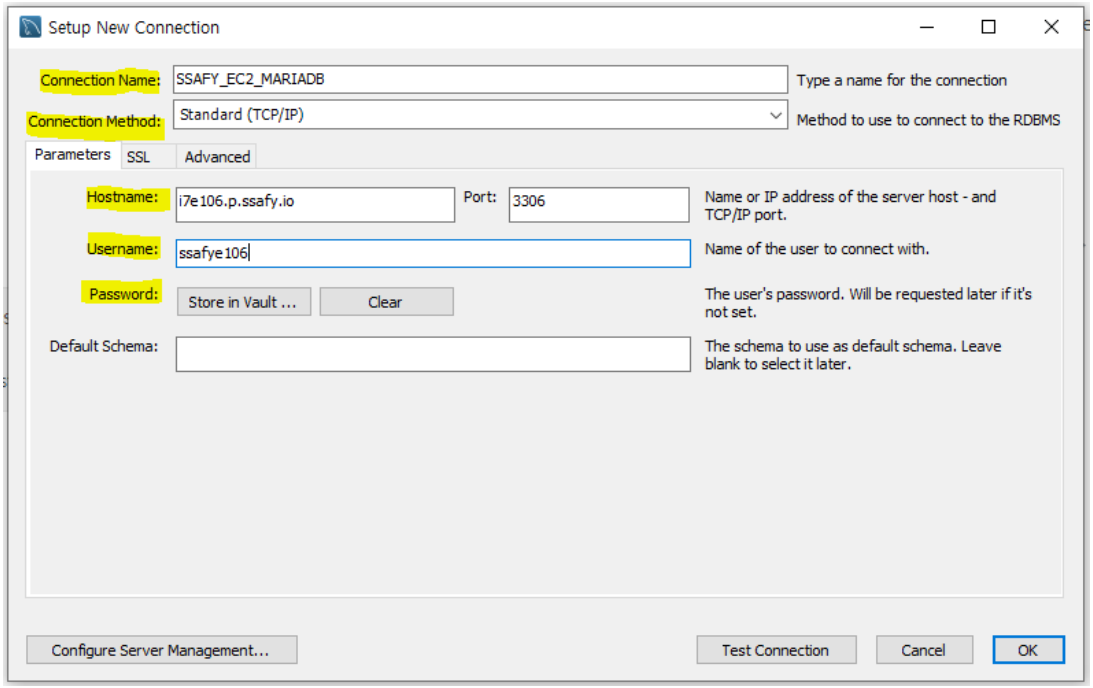
   ※ MariaDB ↔ MySQL 버전 호환 관련 Warning이 뜰 수 도 있는데 무시하고 연결.

5. 연결 테스트가 성공 - OK 누르고 사용

4.2. Standard TCP/IP 연결

1. 홈 화면에서 MySQL Connections 에서 + 추가

2. 아래 사진처럼 설정.



- Connection Name : 본인이 해당 커넥션이 어떤건지 알아보기 쉽게 설정
- Connection Method : Standard TCP/IP 그대로 사용
- Hostname : i7e106.p.ssafy.io 으로 설정 (도메인 이름)
- Username : 계정 ID 으로 설정
- Password : 계정 PW 으로 설정

3. Test Connection으로 연결 확인

   ※ MariaDB ↔ MySQL 버전 호환 관련 Warning이 뜰 수 도 있는데 무시하고 연결.

4. 연결 테스트가 성공하면 OK 누르고 사용

4.3. (공통)Spring Boot에서 연결

- **application.propertis에서 DB 연결 구문을 아래와 같이 수정**

```
spring.datasource.url=jdbc:mysql://[서버도메인]/[스키마 명]?serverTimezone=Asia/Seoul
spring.datasource.username=[계정 ID]
spring.datasource.password=[계정 PW]
```

5. Nginx default

```
server {
  listen 80;
  server_name i7e106.p.ssafy.io;
```

```
  return 301 https://i7e106.p.ssafy.io$request_uri;

}
server {
  listen 443 ssl http2;
  server_name i7e106.p.ssafy.io;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/i7e106.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/i7e106.p.ssafy.io/privkey.pem;

  location / {
    # proxy 설정 (모든 요청 -> 8080으로 전송)
    proxy_pass http://localhost:8080;
    proxy_set_header Host $http_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;

    try_files $uri $uri/ /index.html =404;
  }
}
server {
    if ($host = i7e106.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80;
  server_name i7e106.p.ssafy.io;
    return 404; # managed by Certbot


}
```

6. EC2 Settings

6.1. Docker

1. repository 최신 상태 업데이트 및 HTTP 패키지 설치

```
sudo apt-get update
```

```
sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

2. GPG 키 및 저장소 추가

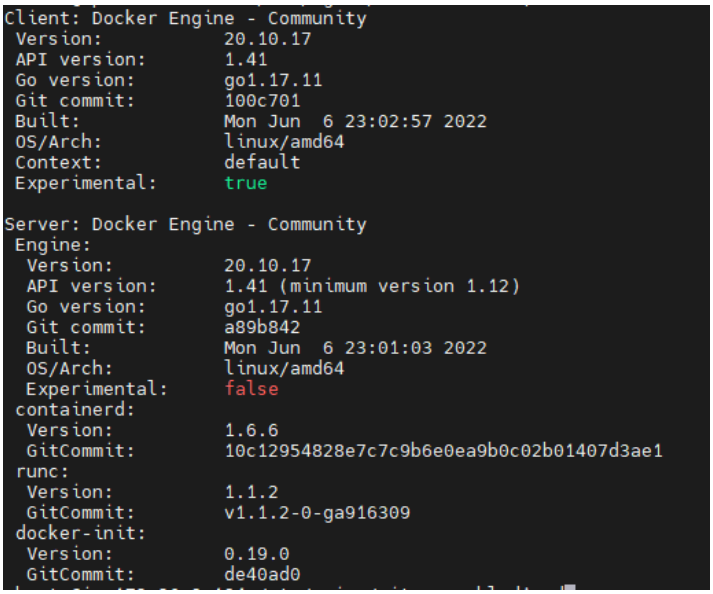
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

3. 도커 엔진 설치

```
sudo apt install docker-ce docker-ce-cli containerd.io
```

4. 도커 버전 확인

```
sudo docker version
```



명령어 실행 화면

6.2. Open Vidu

6.2.1. Install

※ 해당 방식은 공식 문서 (https://docs.openvidu.io/en/stable/deployment/ce/on-premises/) 참조

1. /opt 폴더로 이동

2. git clone으로 설치

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

6.2.2. conf 파일 설정

1. /opt/openvidu폴더(OpenVidu 설치한 폴더)에서 .env 파일 열기

```
sudo vi ./env
```

2. ssl 인증서 관련 설정을 아래와 같이 수정

```
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=[redacted]

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=MY_SECRET

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#                Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#                Please put the certificates files inside folder ./owncert
#                with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#                variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=sonjuhy@gmail.com
```

- **DOMAIN_OR_PUBLIC_IP** : 공인 IP 주소 혹은 도메인 주소 작성
- **OPENVIDU_SECRET** : 클라이언트와 확인 할 비밀 키 값
- **CERTIFICATE_TYPE** : SSL 인증키 적용 방법 선택
  - selfsigned : 자체적으로 가진 인증키 사용(보안 에러 가능성 높음)
  - owncert : 별도의 인증서 키가 이미 있다면 사용. 대신 ./owncert 폴더에 키가 존재 해야함
  - letsencrypt : SSL 인증서 발급 프로그램 이용하여 새로 발급

    ※ 단, LETSENCRYPT_EMAIL을 유효한 이메일로 작성해야함

3. 포트 관련 설정을 아래와 같이 수정

```
HTTP_PORT=80
```

```
HTTPS_PORT=443
```

- 저장 후 편집기 종료

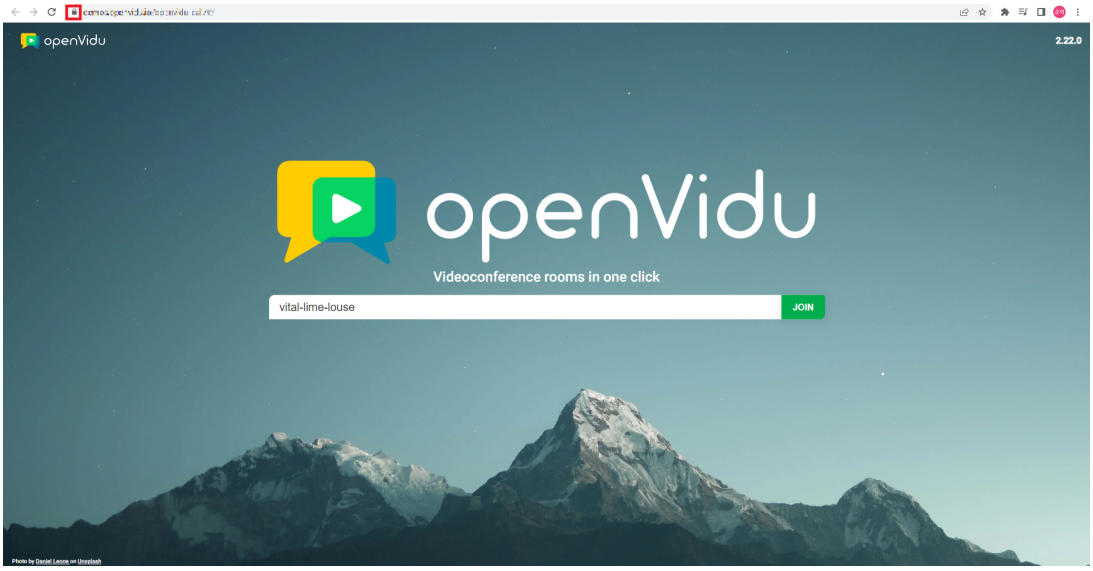
4. CertBot 설치

```
sudo apt-get install letsencrypt -y
```

5. 실행
   a. /opt 폴더에서 아래 명령어로 실행

```
./openvidu start
```

   b. 정상적으로 실행 되는지 확인
      i. 실행한 콘솔 창에 해당 서버 링크와 포트 번호 나오는지 확인
      ii. sudo docker ps 통해 해당 이미지들이 잘 실행 중인지 확인
      iii. 해당 포트로 사이트가 아래 사진처럼 잘 나오는지 확인



특히 주소창 왼쪽 자물쇠 모양(HTTPS 연결) 상태를 꼭 확인 할 것.

iv. **정상 작동 확인이 되었다면 포트를 원하는 포트로 변경 후 재시작**

※ 추후 설치할 Nginx를 위해 변경해야 함.

6.3. Maria DB

6.3.1. Install

1. 아래 명령어를 통해 mariaDB 서버 설치

```
sudo apt-get install mariadb-server
```

2. 서버 설치 끝나면 클라이언트 설치

```
sudo apt-get install mariadb-client
```

3. 슈퍼 계정 및 보안 설정
   a. 아래 명령어로 스크립트 실행

```
sudo mysql_secure_installation
```

   b. 그럼 여러 질문이 나오는데 해당 질문의 뜻과 답변은 아래처럼 진행하면 됨
      1. **현재 root의 패스워드를 입력하시오**

         → 처음 설치시에는 없으므로 **n** 입력
      2. mysql (즉 mariadb) **root 패스워드 설정**할지 선택하시오

         → **y** 눌러서 설정 시작
      3. 원하는 패스워드 입력하시오

→ 원하는 패스워드 입력 후 확인차 한번 더 입력

4. 익명(anonymous) 계정 삭제 여부를 선택하시오

   → **y** 눌러서 삭제

5. root 계정으로 원격 접속 허용 여부를 선택하시오

   → 원격 PC에서 root 계정 통해 접속을 원하면 n, 아니면 y

6. test db를 삭제할건지 선택하시오

   → y 눌러서 삭제

7. 지금까지 설정한 내용을 즉시 반영할지 선택하시오

   → y 눌러서 즉시 반영

c. 설정 완료 하면 mariaDB 접속해서 반영 여부 확인

6.3.2. 계정 생성 및 권한 부여

1. 아래 명령어로 계정 생성

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
```

- username : 계정 이름
- localhost : localhost 로 두면 외부에서 해당 계정 접근 불가능. 외부에서도 사용을 원하면 % 로 작성해야함
- password : 계정 비밀번호

2. 아래 명령어로 해당 계정에 DB(스키마) 접근 권한 설정

```
GRANT SELECT ON db스키마.* TO 'username'@'localhost' identified BY 'password';
```

3. 계정 삭제할 경우 아래 명령어 사용

```
DROP USER ‘username’@’localhost’;
```

6.3.3. 백업 & 복원

1. DB(스키마) 백업

```
sudo mysqldump -u ‘username’ -p ‘password’ db스키마 > backupFileName.sql
```

※ mysqldump는 db에 접속하지 않은 상태에서 실행하는 명령어

2. DB(스키마) 복원

```
sudo mysql -u ‘username’ -p ‘password’ db스키마 < backupFileName.spl
```

```

```

※ DB에 해당 스키마가 존재해야 복원 가능

6.4. Nginx

6.4.1. 설치

1. 아래 명령어로 설치

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

6.4.2. SSL 인증서 발급

1. 아래 명령어로 nginx certbot 툴 설치

```
sudo add-apt-repository ppa:certbot/certbot
```

```
sudo apt-get install python3-certbot-nginx
```

2. **CertBot 실행 - SSL 인증서 발급**
   a. 아래 명령어 실행으로 CertBot 실행

```
sudo certbot --nginx -d 자신의도메인
```

   b. 이후 이메일과 약관 동의 하면 1,2 중 선택 하라고 한다. 각 내용은 다음과 같다.
      i. 1번 선택 : http 요청을 https로 리다이렉션 하지 않는다
      ii. 2번 선택 : http 요청을 https로 리다이렉션 한다.

   c. 선택 후 nginx가 재시작 되면서 https(SSL 인증서) 적용이 완료된다.

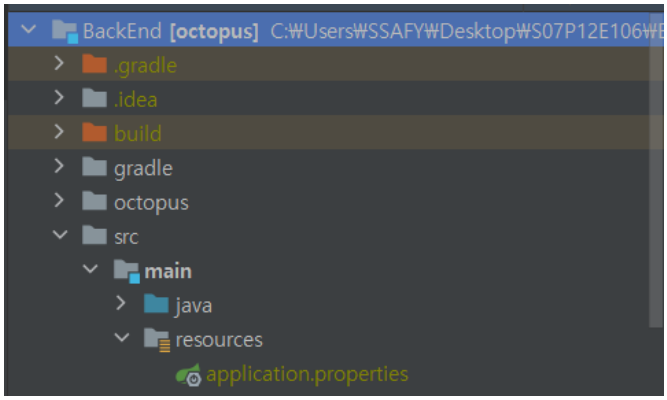
7. Files ignored

1. BackEnd - application.properties

※ DB 계정 등 보안 관련 정보 때문에 별도 관리

a. 파일 위치

- [BackEnd Folder] - src - main - resources - application.properties



b. 파일 내용

- DB(Schma) Auth Info
- JPA Setting
- Swagger

8. etc) Tips

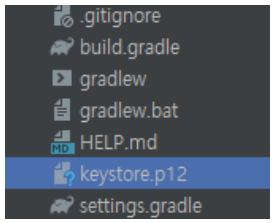
8.1. Windows 개발환경 임시 로컬 ssl 적용

8.1.1. Spring Boot (JAVA) SSL 적용

1. **Spring boot에 https 인증 임시로 localhost에 적용**
   a. 인증서 설치
      i. 실행할 프로젝트 폴더에 아래의 명령어 실행

```
keytool -genkey -alias octopus -storetype PKCS12 -keyalg RSA -keysize 2048 -keystore keystore.p12 -validity 3650
```

      ii. 아래 사진처럼 **keystore.p12**가 생성 되었는지 확인.



   b. 인증서 적용
      i. application.propertis 에 아래 내용 추가

```
## local ssl
server.ssl.enabled=true
server.ssl.key-store=keystore.p12
```

```
server.ssl.key-store-password: [본인이 설정한 패스워드]
server.ssl.key-store-type: PKCS12
server.ssl.key-alias: octopus
```

ii. https://localhost:8080 (혹은 본인이 설정한 포트) 로 잘 작동하는지 확인.

8.1.2. React (JS) SSL 적용

1. 파일 추가
   a. 인증서 설치
      i. Windows 패키지 관리자인 Chocolatey(약칭 : Choco)를 아래와 같이 설치.
         1. cmd 창을 관리자 권한으로 실행.

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy B
ypass -Command " [System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebCli
ent).DownloadString('https://chocolatey.org/install.ps1')) " && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\b
in"
```

            이 내용을 복사 붙여넣기 하여 실행하여 Choco를 설치.

      ii. **관리자 권한으로 실행**한 cmd에 아래와 같이 입력하여 mkcert를 설치.

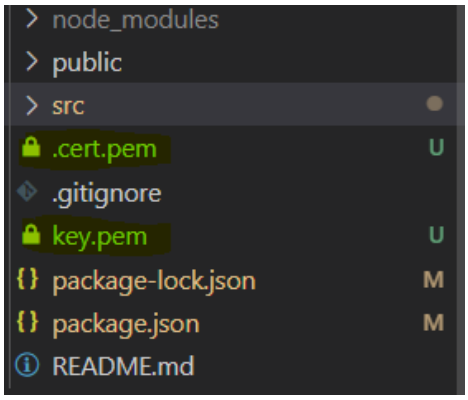
```
choco install mkcert
```

```
mkcert -install
```

      iii. 이후 react가 설치된 폴더에 이동하여 마찬가지로 cmd창에 아래와 같이 입력하여 인증서를 설치.

```
mkcert -key-file ./key.pem -cert-file .cert.pem “localhost”
```

      iv. 아래 사진처럼 파일 두개가 생성되어 있는걸 확인.



생성된 파일 두개 이름 : .cert.pem 그리고 key.pem

b. 인증서 적용

i. package.json 에서 scripts - start를 아래 줄로 변경

```
"start": "set HTTPS=true&&set SSL_CRT_FILE=.cert.pem&&set SSL_KEY_FILE=key.pem&&react-scripts start",
```

ii. 프로젝트 시작 후 적용 확인