목차

1. 기술 스택

가. 이슈관리: Jira

나. 형상관리: Gitlab

다. 커뮤니케이션: Matermost

라. 개발환경

1) OS : Windows 10

2) IDE

가) Spring Tool Suit 3.9.14

나) Visual Studio Code 1.70.1

다) UI/UX : Figma

3) Database : MySQL Workbench 8.0.21

4) Server : AWS EC2 (MobaXterm)

가) Ubuntu 20.04.2

나) Docker 20.10.17

5)

마. 상세사용

1) Backend

가) OpenJDK 1.8.0_192 (Zulu 8.33.0.1)

나) Spring Boot Gradle 7.5

다) Kurento 6.15.0

라) Lombok 1.18.24, Swagger2 3.0.0, QueryDsl 5.0.0

2) Frontend

가) HTML5, CSS3, JavaScript(ES6)

나) Vue 3.2.13, Vuex 4.0.2

다) nodsjs 14.17.0

2. 빌드 상세내용

가. Frontend 빌드

1) front 폴더로 이동

: package.json, jsconfig.json, vue.config.js 등 설정파일이 위치한 곳으로 이동

2) Node_modules를 위한 기본 install

```
npm install
```

3) 빌드하기

```
npm run build
```

back/WebRtc-test/src/main/resources/static 폴더에 빌드 된 파일들이 생성됨

4) 빌드 위치를 바꾸고싶다면...

vue.config.js 파일의 하단 부분을 수정하여 원하는 위치에 빌드

```
module.exports = {
  outputDir: path.resolve(__dirname, '../back/WebRtc-test/src/main/resources/static'),
};
```

나. RestAPI 서버 빌드

1) back/B310_Back 폴더로 이동

: build.gradle, settings.gradle 등 gradle 설정파일이 위치한 곳으로 이동

2) gradle 을 사용한 빌드

: gradle 설치가 되어있지 않다면 설치 후 빌드 진행

가) Windows 10 환경

```
gradlew build --exclude-task test
```

나) ubuntu 환경

: gradle 권한 설정 후, 빌드 진행

```
chmod +x gradlew
./gradlew build --exclude-task test
```

3) 빌드 파일 확인

back/B310_Back/build/libs 위치에

B310_Back-0.3.jar 파일 생성 확인

다. WebRTC 서버 빌드

: 빌드 된 front 파일이 WebRTC 서버 static 폴더에 저장되어 있어 이번 빌드 과정이 통합 빌드 처리가 됨

1) back/WebRtc-test 폴더로 이동

: pom.xml 이 위치한 곳으로 이동

2) maven 을 사용한 빌드

: maven 설치가 되어있지 않다면 설치 후 빌드 진행

```
mvn package
```

3) 빌드 파일 확인

back/WebRtc-test/target 위치에

WebRTC_Server-6.15.0-exec.jar 파일 생성 확인

3. 배포 환경 설정

가. Nginx 설정

1) 기존 Nginx 설정 제거

```
sudo rm /etc/nginx/sites-enabled/default
```

2) Nginx 설정 파일 접근

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

3) 설정 값 다음과 같이 변경

```
# Default server configuration
#
server {
	listen 80 default_server;
	listen [::]:80 default_server;

	root /var/www/html;

	# Add index.php to the list if you are using PHP
	index index.html index.htm index.nginx-debian.html;

	server_name _;

	location / {
		# First attempt to serve request as file, then
		# as directory, then fall back to displaying a 404.
		try_files $uri $uri/ =404;
	}

}

server {

	root /var/www/html;

	# Add index.php to the list if you are using PHP
	index index.html index.htm index.nginx-debian.html;
	server_name i7b310.p.ssafy.io; # managed by Certbot


	location / {
		# First attempt to serve request as file, then
		# as directory, then fall back to displaying a 404.
```

```
                proxy_pass https://localhost:8443;
                proxy_set_header Origin "";
                proxy_http_version 1.1;
                #proxy_set_header Upgrade $http_upgrade;
                #proxy_set_header Connection $connection_upgrade;
                proxy_set_header Upgrade "websocket";
                    proxy_set_header Connection "Upgrade";
                proxy_read_timeout 120000; # 2 * 60 * 1000
                proxy_send_timeout 120000; # 2 * 60 * 1000

                    proxy_set_header X-Real-IP $remote_addr;
                    proxy_set_header                    X-Forwarded-For
$proxy_add_x_forwarded_for;
                    proxy_set_header Host $http_host;

                #try_files $uri $uri/ =404;
        }

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate    /etc/letsencrypt/live/i7b310.p.ssafy.io/fullchain.pem;    #
managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/i7b310.p.ssafy.io/privkey.pem; #
managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam    /etc/letsencrypt/ssl-dhparams.pem;    #    managed    by
Certbot

}
server {
        if ($host = i7b310.p.ssafy.io) {
            return 301 https://$host$request_uri;
        } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
        server_name i7b310.p.ssafy.io;
        return 404; # managed by Certbot
```

```
}
```

4) 작성한 파일 적용

: sites-enabled 에 같은 파일이 옮겨지고 nginx 를 재실행 하면 설정이 적용됨

```
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

5) nginx 재실행

```
sudo service nginx restart
```

나. DB 설정

1) MySQL 접속

: MySQL이 설치되지않았다면 설치 후 진행

```
sudo mysql -u root -p
```

* 비밀번호 입력의 경우 초기에는 없거나 'root'

2) 로컬 계정 생성

: 설치한 곳에서만 사용 가능한 계정

```
create user 'ssafy'@'localhost' identified by 'ssafy';
```

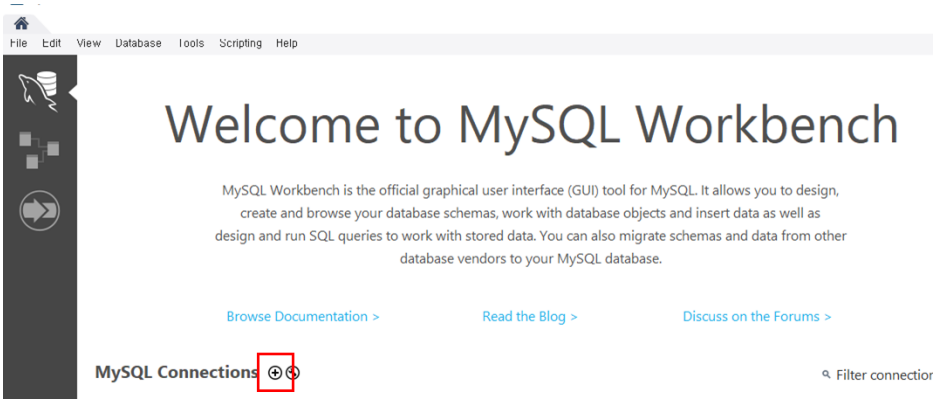
3) 외부 접근 계정 생성

: 외부에서 DB를 관리할 때 사용할 계정

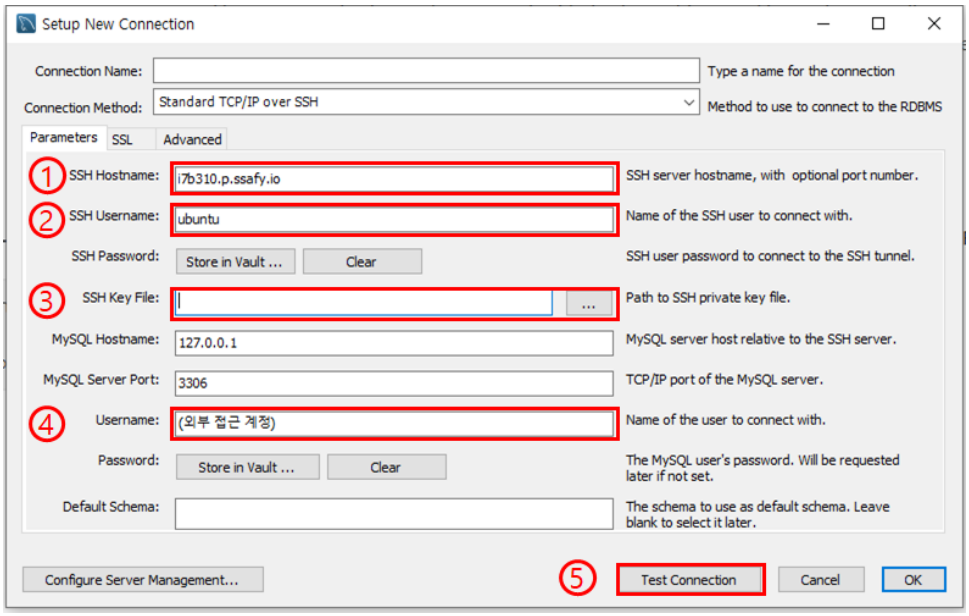
```
create user '(사용할 계정)'@'%' identified by '(사용할 비밀번호)';
```

4) Workbench로 서버 DB 연결

가) 새 연결 추가 버튼 클릭



나) 연결에 필요한 값 입력



① SSH Hostname 입력 (EC2 주소 @ 뒷부분)

② SSH Username 입력 (EC2 주소 @ 앞부분)

③ SSH Key File 파일 경로 입력 (EC2 의 .pem 키 위치)

④ Username 입력 (생성한 외부 접근 계정)

⑤ Test Connection 클릭 후 접속 확인

⑥ OK 눌러 등록

다) 새로운 연결 등록 확인

아래와 같은 연결 블록이 생겼다면 성공

5) 기본 더미 데이터 입력

exec/dummyDB.sql 파일 사용하여 더미데이터 SQL 실행

다. 기본 더미 데이터 파일 입력

exec/img.zip 파일 압축 해제하여

/home/ubuntu/ 위치에 더미 폴더가 저장된 img 폴더 옮기기

라. Kurento Media Server

1) Docker 설치

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

```
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
```

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

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

2) Kurento Media Server 이미지 받기

```
sudo docker pull kurento/kurento-media-server:latest
```

3) Kurento Media Server 컨테이너 실행

: kms 라는 이름으로 기본 설정포트(8888) 로 실행

```
sudo docker run -d --name kms --network host ₩
kurento/kurento-media-server:latest
```

4) Coturn Server 설치

: WebRTC 시그널링을 위한 STUN/TURN 서버

가) 설치

```
sudo docker run -d --name kms --network host ₩
kurento/kurento-media-server:latest
```

나) 설정

A. /etc/default/coturn 파일 수정

```
TURNSERVER_ENABLED=1
```

B. /etc/turnserver.conf 파일 수정

```
listening-port=3478
tls-listening-port=5349
listening-ip=(EC2 Private IPv4)
external-ip=(EC2 Public IPv4)/(EC2 Private IPv4)
relay-ip=(EC2 Private IPv4)
fingerprint
lt-cred-mech
user=myuser:mypassword
realm=myrealm
log-file=/var/log/turn.log
simple-log
```

다) Coturn 재 실행

```
sudo service coturn restart
```

4. 배포 명령어

가. EC2에 빌드한 2개의 jar 파일 업로드

나. 배포 전 구동 중인 서버 확인

: 서버의 포트는 8081, 8443 으로 설정되어있음

1) 8081, 8443 포트로 실행중인 서버 PID 확인

```
netstat -ntlp | grep :8081
netstat -ntlp | grep :8443
```

2) 실행 중인 PID 가 있을 경우 종료

```
sudo kill <PID>
```

다. 서버 실행

```
nohup java -jar (실행할 빌드 파일.jar) &
```