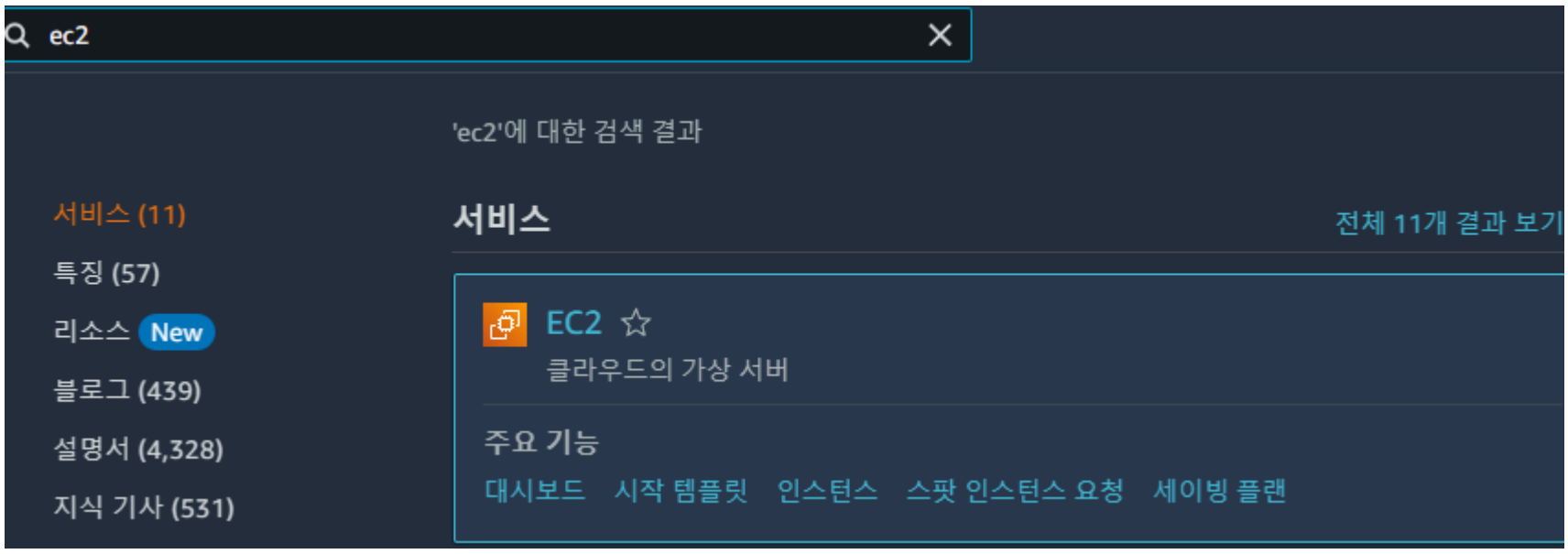
이번 목표는 프로젝트를 AWS의 **EC2 서버**에서 **배포**하는 것이다.

# AWS EC2

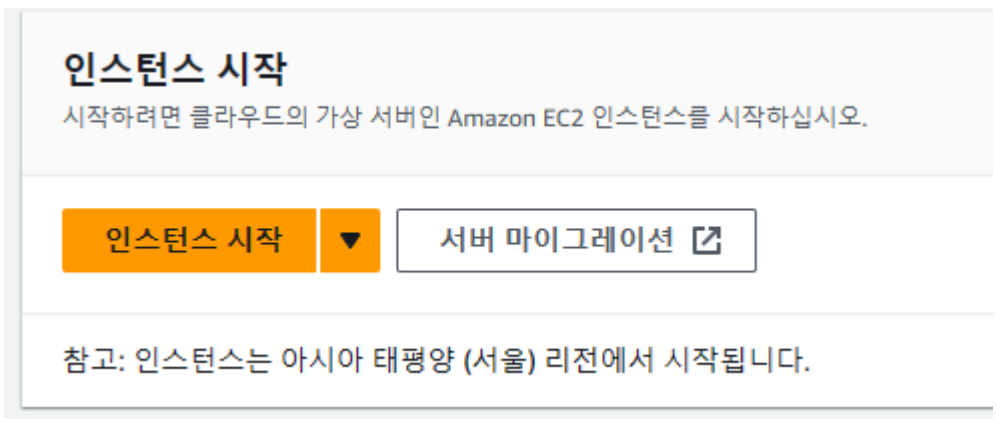
## 인스턴스 생성

> **AWS(Amazon Web Services)**는 **클라우드** 컴퓨팅 플랫폼으로, **인스턴스**라는 **가상 서버**를 생성해 사용할 수 있다. **EC2(Elastic Compute Cloud)**는 AWS가 제공하는 여러 서비스 중 하나이며, **탄력적**(Elastic)이라는 이름에 맞게 사용자 필요에 따라 인스턴스를 확장하거나 축소할 수 있다.

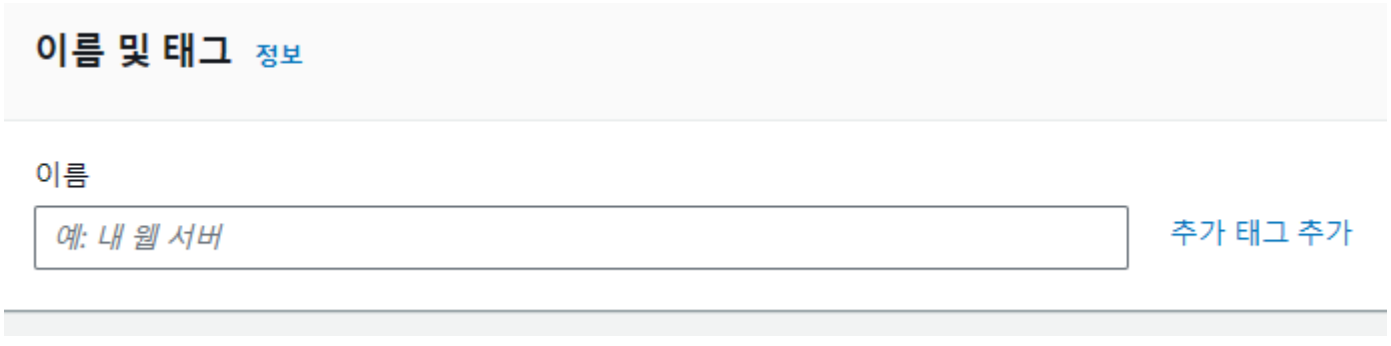
EC2 인스턴스 생성하는 부분은 다른 좋은 자료가 많으니 **대충**만 짚고 넘어간다.



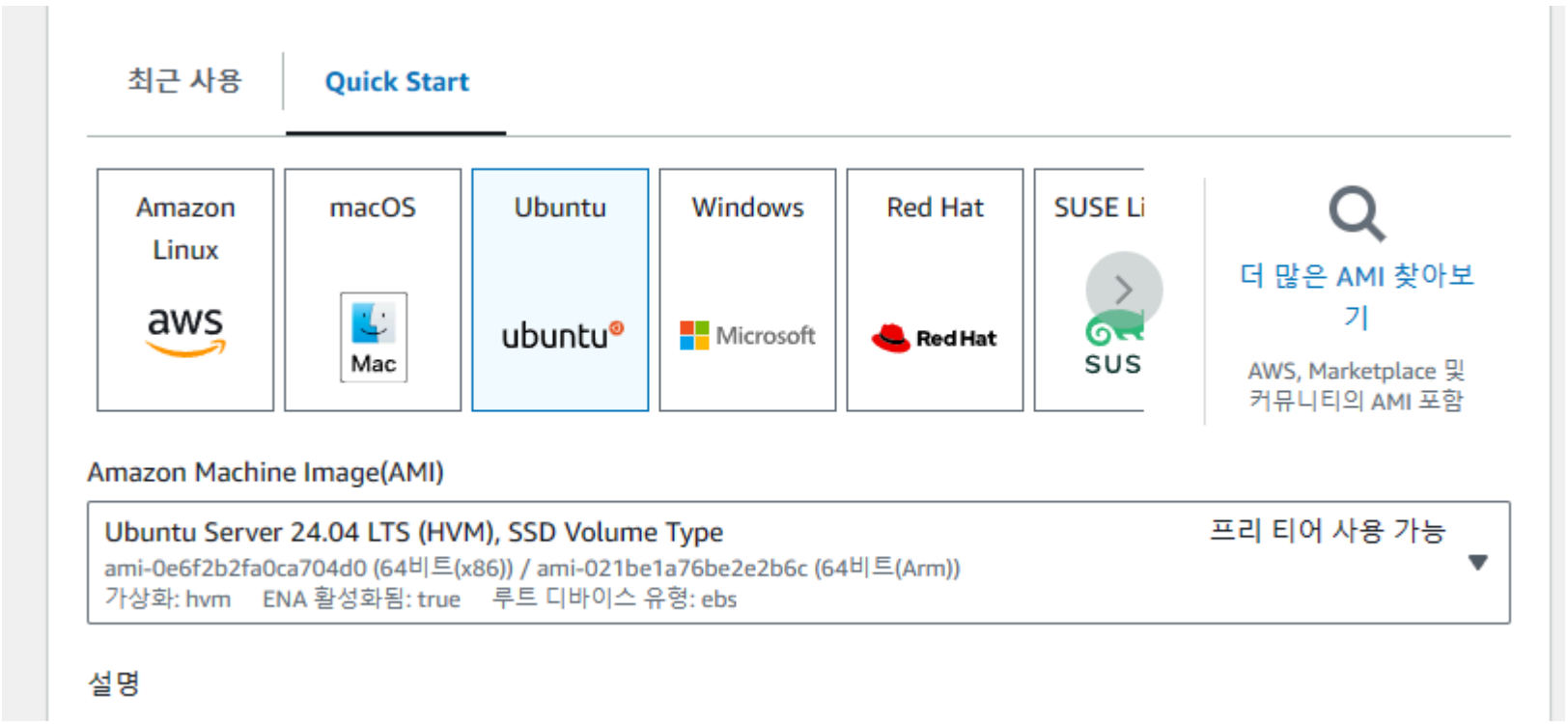

AWS 가입하여 콘솔에 접속한 후 ec2를 검색하고




인스턴스 시작을 누른다.

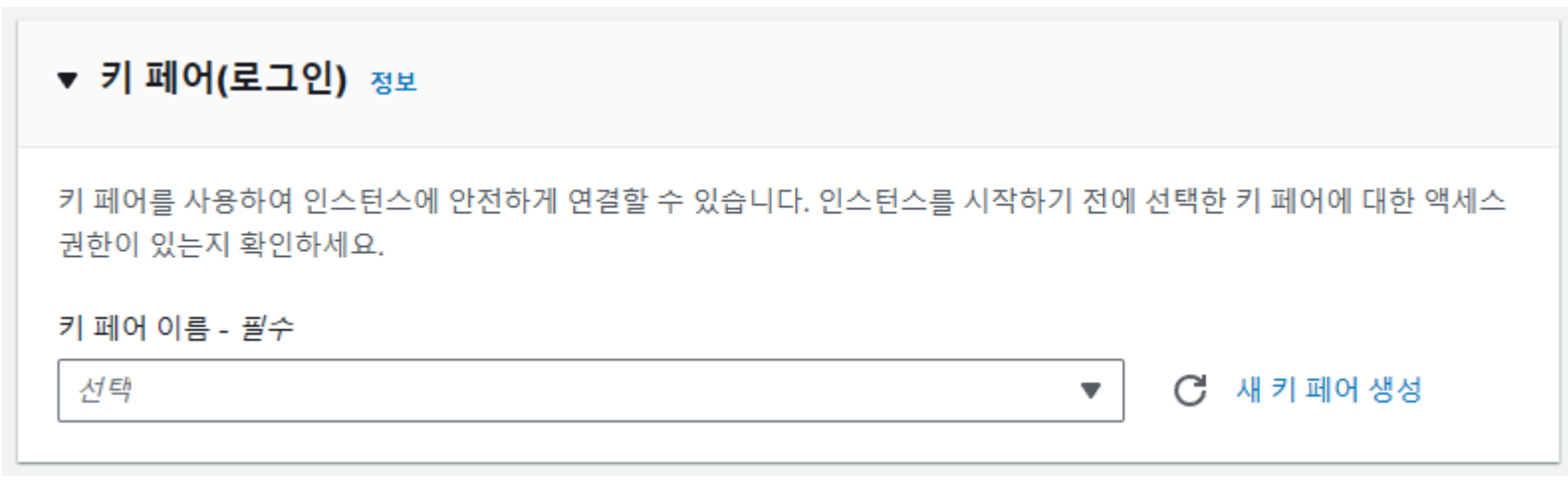



인스턴스 이름을 정하고,



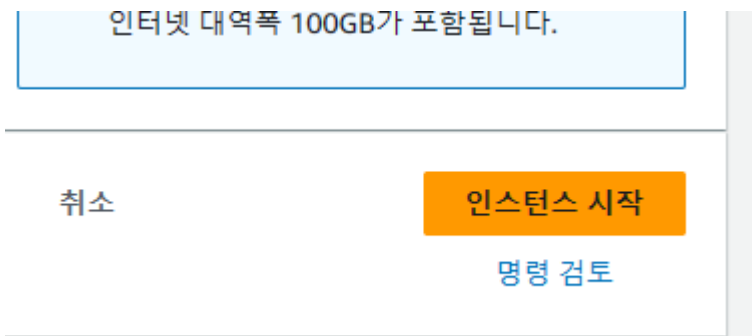

운영체제로는 **우분투(Ubuntu)**를 선택한다.


인스턴스 유형으로 **프리 티어**에서 사용 가능한 `t2.micro` 를 사용한다.

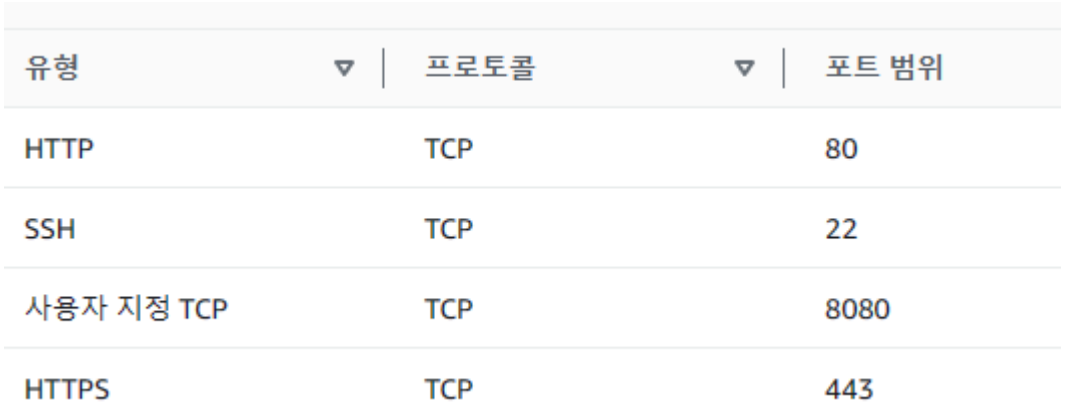



서버 접속에 사용할 **키 페어**. 기존 것을 사용할 생각이 아니라면 새로 생성한다.


**키 페어** 이름을 적고 키 페어 생성 버튼을 눌러 키 페어를 다운받는다.
이 키가 없으면 서버 접속이 불가능하니 안전하게 보관한다.
방금 키 페어 칸에서 생성한 키를 선택한다.




마지막으로 **인스턴스 시작**을 눌러 인스턴스 생성을 마친다.



| 유형 | 프로토콜 | 포트 범위 |
| --- | --- | --- |
| HTTP | TCP | 80 |
| SSH | TCP | 22 |
| 사용자 지정 TCP | TCP | 8080 |
| HTTPS | TCP | 443 |

이후 필요에 따라 **보안그룹** 등을 수정한다.

## 비용 관련

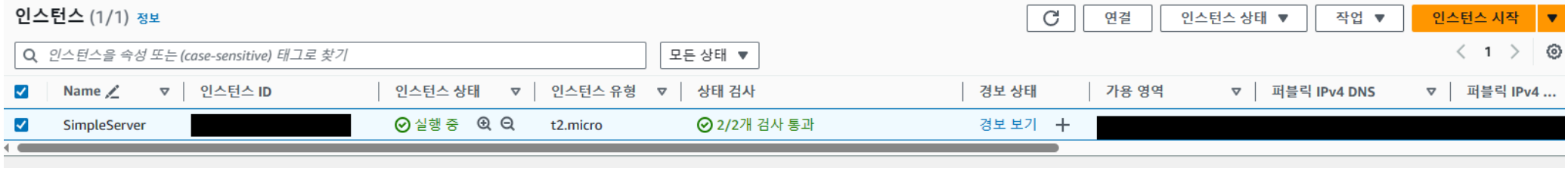
AWS 계정을 활성화하면 **1년** 간 지정된 한도 내에서 여러 서비스를 **무료**로 사용할 수 있는 **프리 티어**를 사용할 수 있다. 자칫하다 까먹으면 요금이 청구될 수 있으니 사용하지 않는 인스턴스는 정지하거나 삭제해 두는 것이 좋다.

생성된 인스턴스의 **퍼블릭 IPv4 주소**는 인스턴스의 상황에 따라 **변한다**. **탄력적 IP**를 발급받아 인스턴스에 배정하면 주소를 고정시킬 수 있다. 하지만 **요금이 발생**할 수 있다. *(퍼블릭 IPv4 주소도 이제 요금이 발생한다고 한다. 프리 티어는 무료 이용 가능한 시간 제공)*

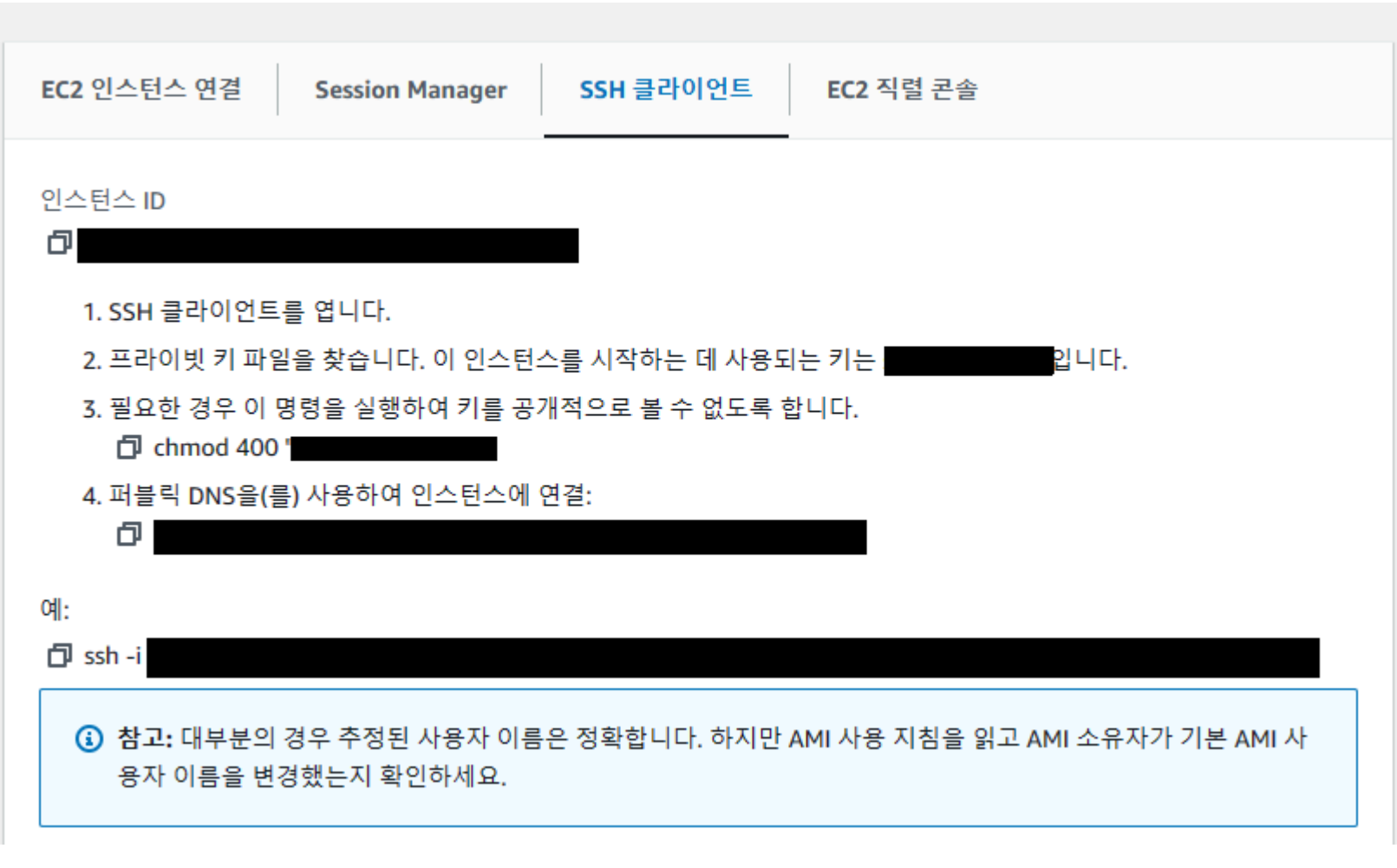
~~연습하고 귀찮아서 방치했다가 요금 나온적있음~~

# 인스턴스 접속

이제 생성한 인스턴스에 **접속**해 보자.




인스턴스 목록에서 생성한 인스턴스를 선택하고 오른쪽 위 **연결** 버튼을 클릭한다.




여러 접속 방법이 써있다. 여기에선 `SSH 클라이언트` 탭에 표기된 방법을 사용할 것이다.

다운받은 **키**가 있는 폴더로 가 **터미널**을 열고 3번 지시문에 써져있는 명령어를 *복사 붙여넣기*한다.
키의 **권한**을 **제한**하기 위함인데, 설정하지 않으면 접속을 시도할 때 키 권한이 너무 열려있다고 **접속이 안된다**.

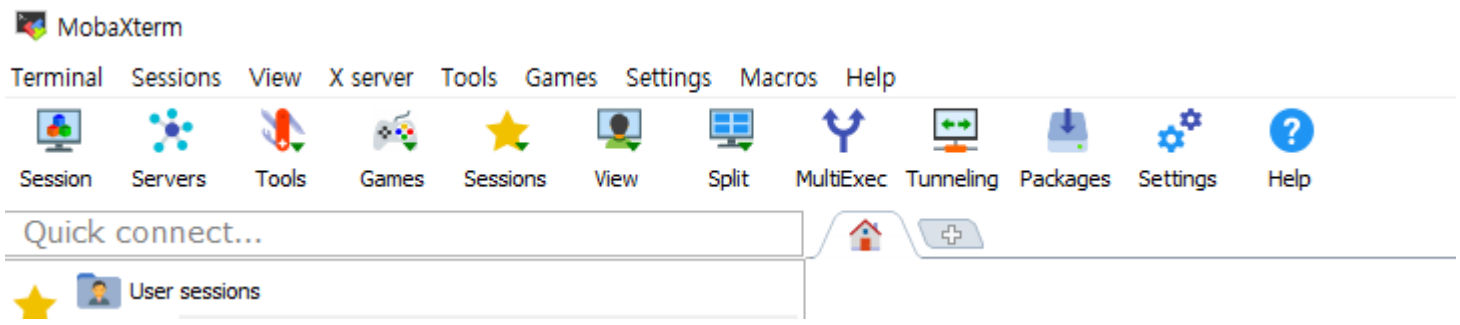
만약 모종의 이유로 권한 설정이 잘 안됐다면 *직접* 키 파일을 우클릭 해, *속성 - 보안탭* 쪽을 수정해야 한다. 자세한 내용은 검색.

권한 수정에 성공했다면 마지막 명령문을 복사 - 붙여넣기하여 실행하면 **접속 성공**이다. (키가 있는 폴더에서)

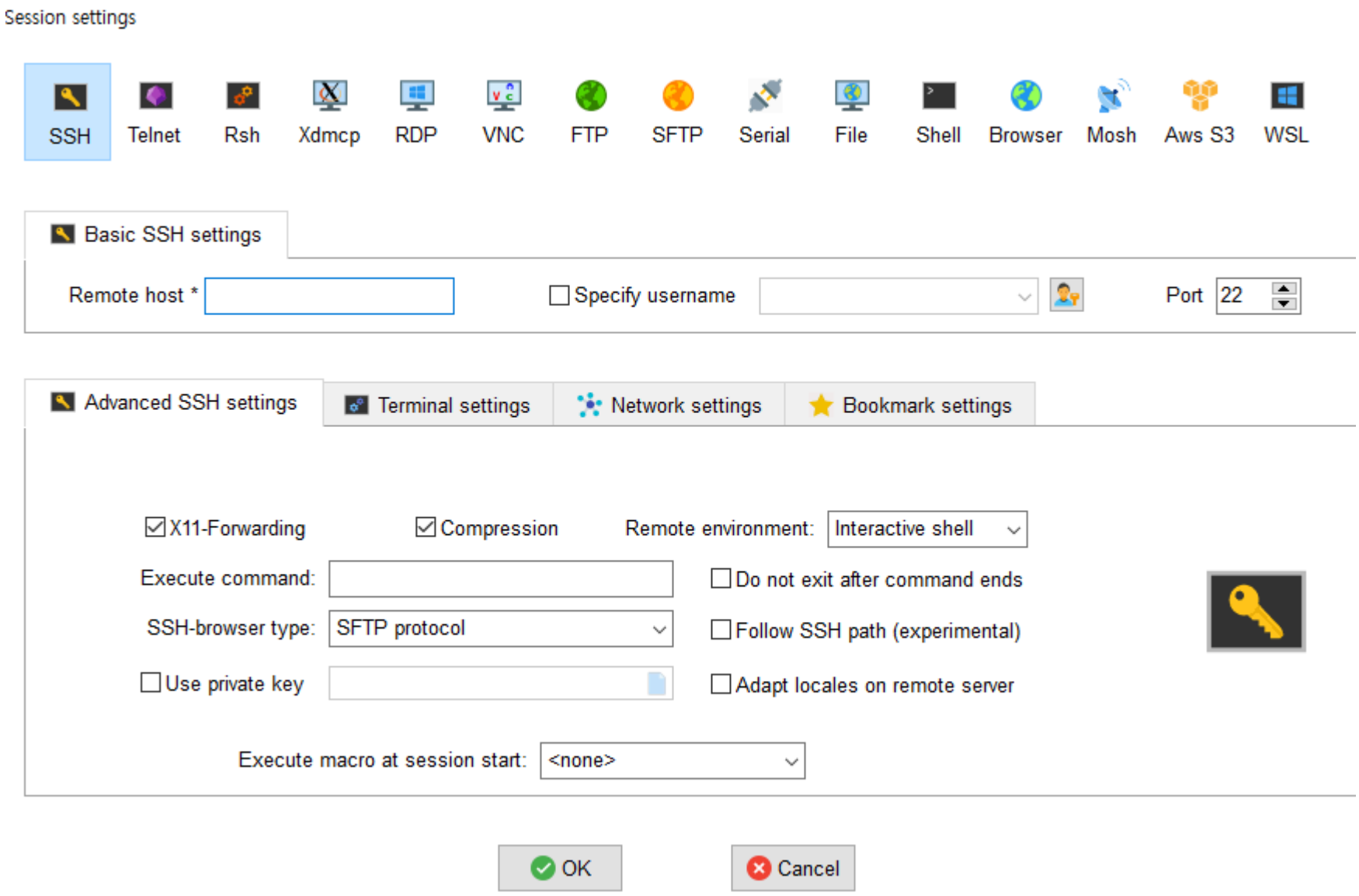
# MobaXTerm으로 서버 접속

매번 터미널에서 명령어를 사용해 접속해도 되지만 *Putty, MobaXTerm, Terminus* 같은 **프로그램**을 사용하면 더 편리하다.

그 중 **MobaXTerm**을 사용했기 때문에 그 방법에 대해서 간략하게 적겠다.




프로그램을 설치하고 왼쪽 위 **Session**을 클릭한다.



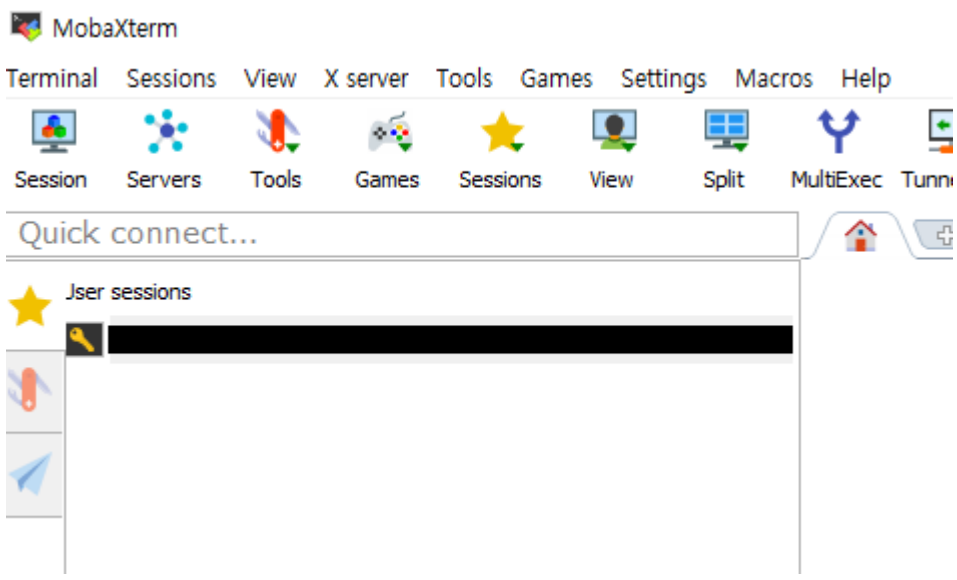

나온 창에서 **SSH** 탭 - **Adavanced SSH settings** 탭을 연달아 클릭하면 다음과 같은 화면이 나온다.

왼쪽 위 **Remote host** 칸에는 인스턴스의 **퍼블릭 IPv4 주소**를 입력한다. 이는 AWS 웹 페이지에 들어가 인스턴스의 세부 정보 부분에서 확인할 수 있다.

오른쪽 위에 **Specify username**에 체크하고 사용자 이름을 입력한다. 인스턴스의 OS는 우분투고, 우분투의 기본 사용자 이름은 `ubuntu` 이므로 이를 입력한다.

왼쪽 아래에서 **Use private key**를 체크하고 다운받았던 **키**를 등록한다.

마지막으로 OK를 누르면 세션이 만들어진다.



이후부턴 프로그램을 실행하고 원하는 세션을 클릭하면 바로 접속이 된다. *(인스턴스의 주소가 달라지는 경우 새로 등록)*

# 서버에 프로젝트 업로드

EC2 인스턴스를 생성했으면 서버로 프로젝트를 옮겨야 한다.
방법은 다음 두 가지가 있다.

**1) Github**을 통해 프로젝트를 통째로 인스턴스로 옮긴 후, **jar** 파일 생성함.
**2)** 로컬에서 **jar** 파일 생성 후, 인스턴스로 옮김.

mobaXTerm이 설치되어 있고, 단순 연습용이라 깃헙 연동 등이 귀찮다면 2번 방법이 편하다.

> **JAR(Java ARchive)** 파일은 여러 *자바 클래스* 파일과 클래스들이 이용하는 관련 리소스(텍스트, 그림 등) 및 메타데이터를 하나로 **압축**시킨 파일이다. **자바 애플리케이션**을 **배포**하고 **실행**시키기 위해 사용한다.

## 방법 1.

```
$ sudo apt-get install git
```

인스턴스에 접속해 다음 명령어를 입력하여 Git을 설치한다. (Ubuntu는 apt-get, Linux는 yum)

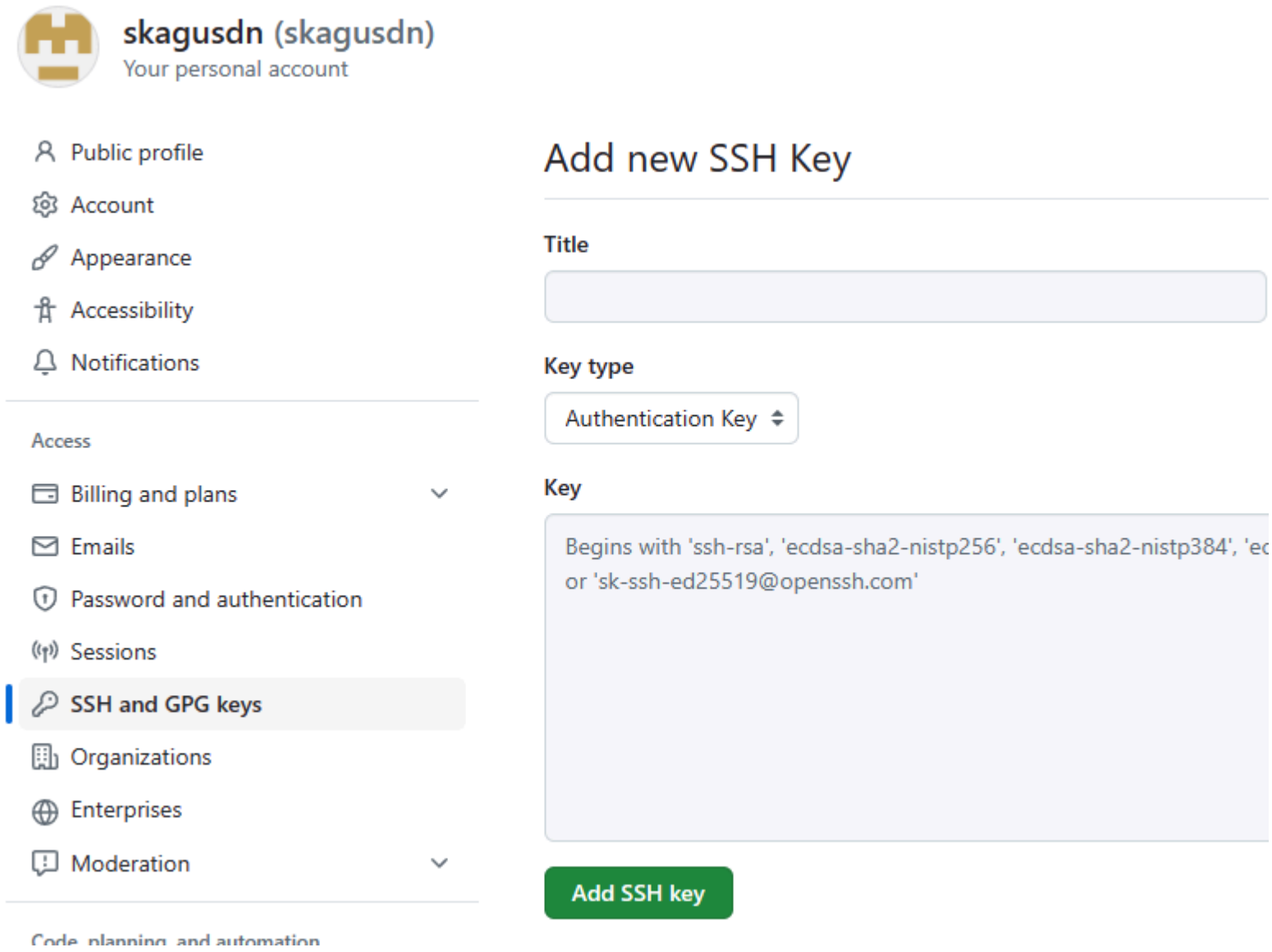
```
$ cd ~/.ssh
$ ssh-keygen -t rsa -C [Github 계정 이메일]
```

다음 명령어를 통해 **ssh 키**를 생성한다.
***키 생성 이유*** *: 키를 생성해서 등록해 놓으면 git을 쓸 때 마다 아이디, 비밀번호 안 써도 됨.*

`ssh-keygen` : ssh 키를 생성하는 프로그램.
`-t` 키를 생성하는 방식에 대한 옵션. `rsa` 는 그런 방식 중 하나.
`-C` 주석 옵션. Github에서 이메일을 넣어놓으라 가이드함. 필수인지는 잘..

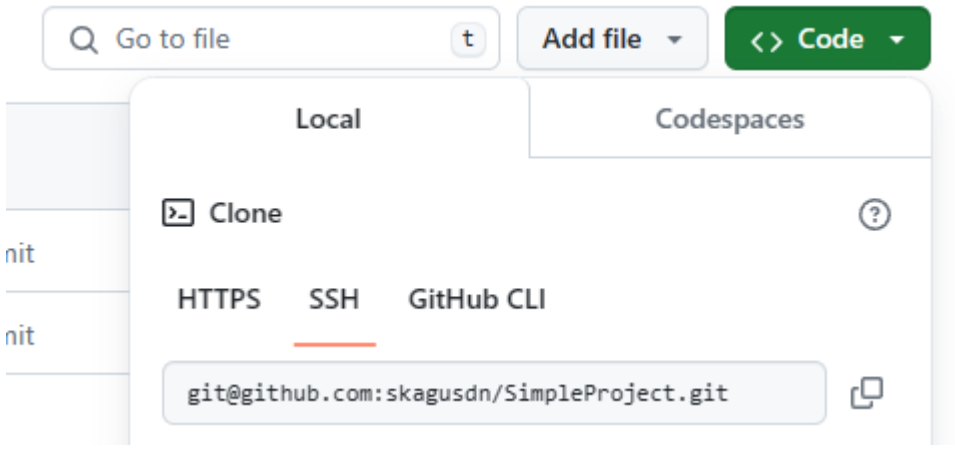
```
$ cat id_rsa.pub
```

생성된 `id_rsa.pub` 파일의 내용물을 다음 명령문을 실행하여 출력한다.
출력된 내용을 **복사**해둔다.




이후 *Github 로그인 - 프로필 클릭 - Settings - SSh and GPG keys* 에서

오른쪽 위 **New SSH Key** 버튼을 누르고 복사한 내용물을 아래에 붙여넣기 한다.
키 이름도 지어준 후 **Add SSH key** 버튼을 눌러 생성을 마친다.




이후 Github에 프로젝트를 올린 후, 인스턴스에서 `git clone [주소]` 을 통해 프로젝트를 복사한다. 위 사진처럼 사용할 주소는 **Code - SSH**에 있다. 일반적으로 사용하는 주소와 다르다는 것에 유의.

```
./gradlew build
```

프로젝트 디렉토리로 가서 다음 명령어를 통해 빌드를 하면 `/build/libs` 디렉토리에 **jar** 파일이 생성된다.
*(프로젝트 빌드 도구로 gradle을 선택했을 때 명령어. maven을 택했다면 다름.)*
*(gradle 버전에 따라 비슷한 이름에 plain이 붙은 jar 파일이 생성될 수 있다. 의존성을 포함하지 않아 실행되지 않기 때문에 무시)*

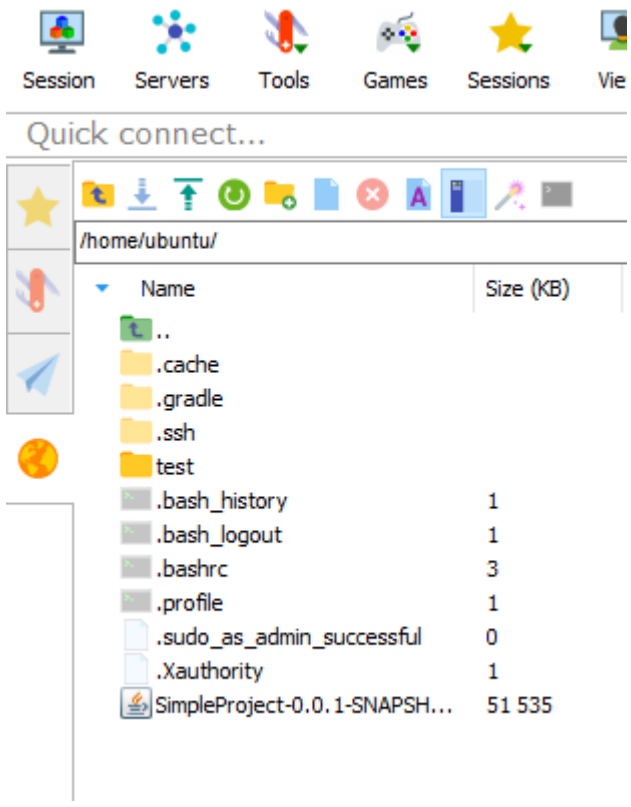
```
chmod +x gradlew
```

만약 에러가 난다면 위 명령어로 `gradlew` 의 권한을 변경한 후 다시 해 보자.

# 방법 2.

```
./gradlew build
```

로컬 쪽 프로젝트 디렉토리에서 이 명령어를 실행하면 마찬가지로 `/build/libs` 에 **jar** 파일이 생성된다.




생성된 jar 파일을 인스턴스로 옮긴다. *mobaXTerm* 등 프로그램으로 서버에 접속했다면 파일을 드래그해서 원하는 폴더에 두는 것만으로 파일을 옮길 수 있다. (딸깍! - 끝!)

# 실행

이제 프로그램을 실행시켜 보자.

# 메모리 늘리기

필수X.

무료로 낮은 사양의 서버를 사양하다 보니 조금만 무거운 작업을 시켜도 서버가 뻗는다.
진짜 메모리를 늘리는 것은 아니고 **메모리 스왑**을 통해 디스크 공간을 메모리처럼 사용하는 **가상메모리** 기법이다.

링크 : https://repost.aws/ko/knowledge-center/ec2-memory-swap-file

AWS 사이트의 가이드를 따라하기만 하면 된다. (가이드에선 4기가. 서버 디스크 용량에 따라 늘릴 것.) (혹시 몰라서 인스턴스 재부팅함.)

# Java 설치

이제 필요한 Java와 MySQL을 설치해보자.

```
// 설치 가능한 리스트 최신화
$ sudo apt update

// 패키지를 최신 버전으로 업그레이드
$ sudo apt upgrade
```

위 명령어로 ubuntu 패키지를 업데이트 한 후,


```
$ sudo apt list openjdk*
```

다음 명령어로 설치 가능한 `openjdk` **버전 목록**을 둘러 본다.

```
$ sudo apt list openjdk-21-jdk
```

목록을 참고해서 원하는 Java **버전**을 설치한다.
이 프로젝트는 openjdk-21 버전이 필요하므로 다음과 같은 명령어로 설치했다.

# MySQL 준비

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

다음 명령어로 MySQL를 **설치**한다.

```
$ sudo ufw allow mysql
```

원격 db 연결이 필요하다면 다음 명령어를 추가한다.

```
$ mysql -u root
```

MySQL에 **접속**한 후,

```
mysql> CREATE USER '[유저이름]'@'localhost' IDENTIFIED BY '[비밀번호]';
mysql> GRANT ALL PRIVILEGES ON *.* TO '[유저이름]'@'localhost' WITH GRANT OPTION;
```

다음 명령어로 MySQL **사용자**를 만든다.
물론 프로젝트의 설정 파일에 적힌 것과 **일치**해야한다.

아래 명령문은 해당 유저에게 모든 데이터베이스에 대해 모든 권한을 부여한다.

```
mysql> CREATE DATABASE [DB이름];
```

사용할 **데이터베이스**도 만든다.
이것도 설정 파일에 쓰인 이름과 일치하게.

> *MySQL은 **윈도우**에선 **대소문자**를 구분하지 않지만, **리눅스**에선 구분한다. 이거 몰라서 왜 오류나나 한참 고생했다.*

# Spring Boot 어플리케이션 실행

이제 드디어 실행 준비가 끝났다.

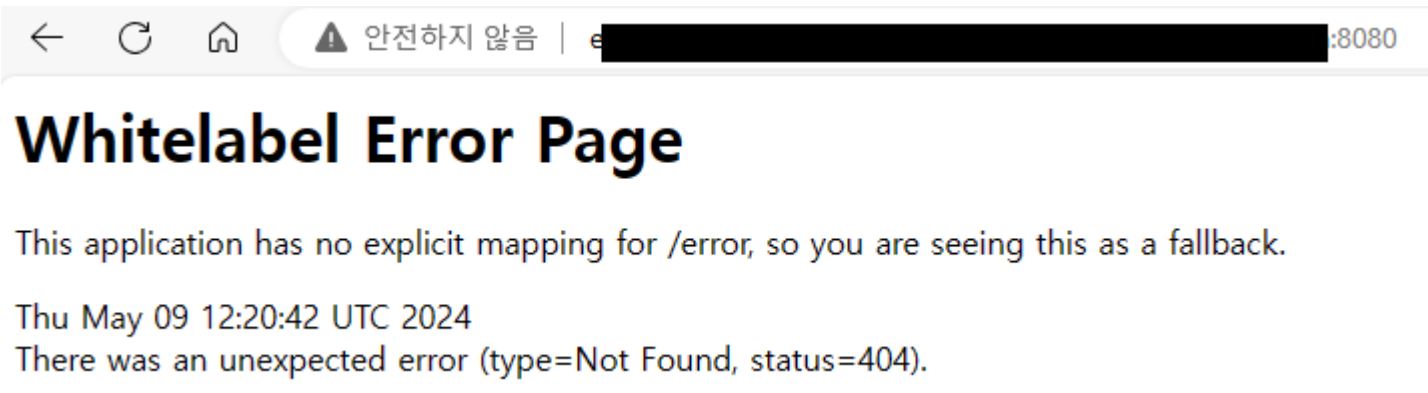
```
$ java -jar [jar 파일 이름]
```

**jar** 파일이 있는 디렉토리로 와 다음 명령어를 실행한다.

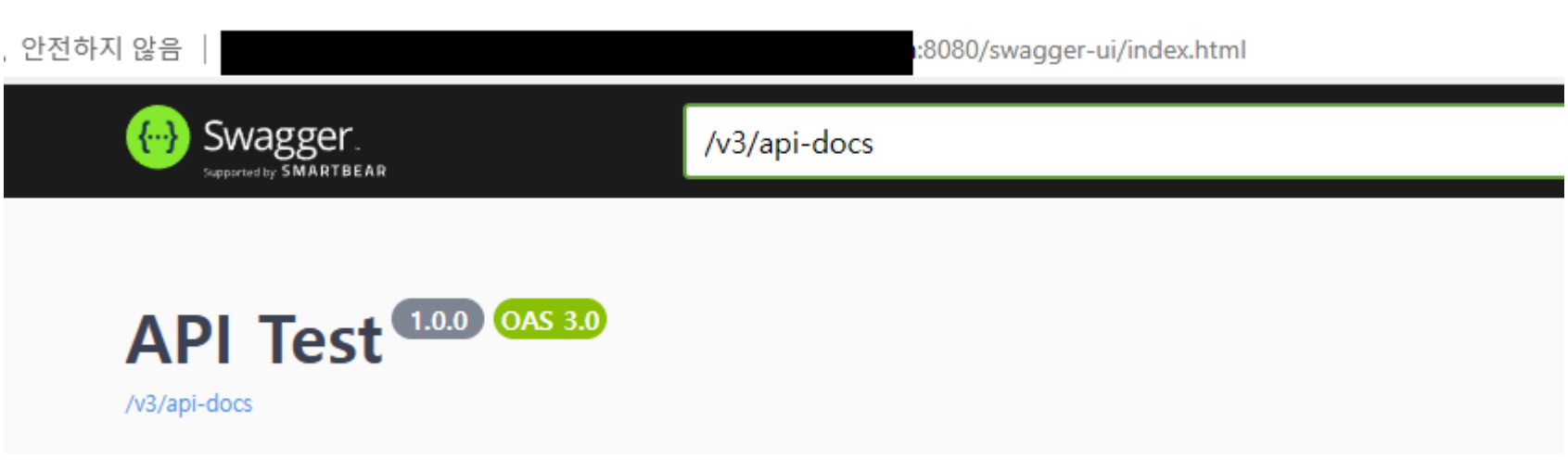
```
$ nohup java -jar [jar 파일 이름] &
```

- **nohup** :
  앞에 붙이면 터미널 접속을 끊어도 프로그램이 중단되지 않고 계속 실행됨. 프로그램 출력이 `nohup.out` 파일에 저장됨.
- **&(백그라운드)** :
  백그라운드로 프로그램을 돌리게 하는 명령어. 원래는 *nohup*과 달리 터미널 접속을 끊으면 프로그램이 중단되지만 기본 옵션이 바껴서 *nohup*과 유사하게 동작한다고 함.


잘 되었다면 우리가 많이 본 이 화면이 등장할 것이다.




인스턴스의 **퍼블릭 IPv4 주소** 끝에 `:8080` 을 붙여 접속을 시도했을 때 다음과 같은 ㅎ 나면이 나오면 성공이다.




만약 swagger도 설정해 놓았다면 접속이 되는 것도 확인할 수 있다.

```
mysql> use simpledb;
Reading table information for comple
You can turn off this feature to get

Database changed
mysql> show tables;
+--------------------+
| Tables_in_simpledb |
+--------------------+
| simpleentity       |
| simpleentity_seq   |
+--------------------+
2 rows in set (0.00 sec)

mysql> select * from simpleentity;
+----+---------------+
| id | name          |
+----+---------------+
|  1 | "IS THIS OK?" |
+----+---------------+
1 row in set (0.00 sec)
```

swagger를 통해 DB에 데이터 넣는 것도 잘된다!