

스프링 부트 3



세상에 선보이는 **SBB 서비스!**



4-01 이제 서버가 필요하다!

4-02 AWS 라이트세일 알아보기

4-03 서버 접속 설정하기

4-04 서버 접속 프로그램 설치하기

4-05 SBB 배포하기

4-06 서버 스크립트 생성하기

4-07 서버 환경으로 분리하기

4-08 80번 포트로 웹 서비스 운영하기

4-09 로그 관리하기

4-10 도메인 사용하기

4-11 HTTPS로 전환하기

4-12 PostgreSQL로 전환하기

클라우드 시스템

- 서버/운영체제/데이터베이스/네임 서버 제공과 관리해줌
- 아마존 웹 서비스(Amazon Web Services), 즉 AWS를 사용함.

AWS 라이트세일이란?

- AWS 라이트세일은 아마존에서 운영하는 웹 서비스에 특화된 클라우드 서비스
- AWS 라이트세일은 웹 서비스 운영에 꼭 필요한 기능만 준비되어 있음
- 첫 3달은 무료이며 그 이후에는 매달 5달러씩 사용료 지불

AWS 가입하기

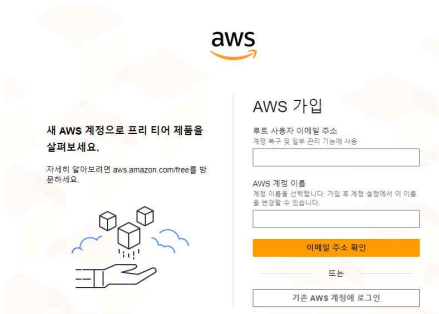
먼저 AWS 공식 홈페이지(<https://aws.amazon.com/ko>)에 접속한 다음,
[AWS 계정 생성]을 클릭함



AWS 가입하기

다음과 같은 AWS 계정 생성 화면이 나타남.

이 화면에서 이메일 주소와 계정 이름을 입력하고 [이메일 주소 확인]을 클릭함



The image shows the AWS sign-up page. At the top is the AWS logo. Below it, on the left, is a promotional message about free tier products. On the right, under the heading 'AWS 가입', are input fields for email and account name, followed by a 'Verify email address' button and links for 'or' and 'Sign in to your existing AWS account'.

aws

새 AWS 계정으로 프리 티어 제품을
살펴보세요.

자세히 알아보려면 aws.amazon.com/free를 방문하세요.

AWS 가입

루트 사용자 이메일 주소
계정 복구 및 일부 관리 기능에 사용

AWS 계정 이름
계정 이름을 선택합니다. 가입 후 계정 설정에서 이 이름을 변경할 수 있습니다.

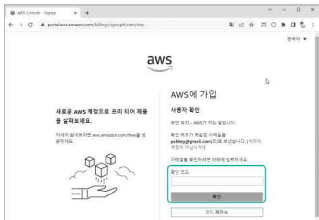
이메일 주소 확인

또는

기존 AWS 계정에 로그인

AWS 가입하기

1. 클릭한 후, 다음과 같은 '사용자 확인' 화면이 나타남.
등록한 이메일로 전달된 확인 코드를 입력한 후 [확인]을 클릭함



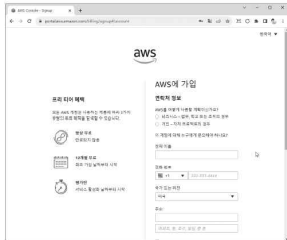
AWS 가입하기

클릭한 후, 다음과 같이 암호를 입력하는 화면이 나타남.
암호를 입력하고 [계속]을 클릭함



AWS 가입하기

5. 그러면 다음과 같은 '연락처 정보' 화면이 나타남.
이름, 전화번호, 주소 등을 입력한 후, '동의'에 체크하고 [계속]을 클릭함.
이때 이름과 주소는 반드시 영문으로 입력해야 함



The screenshot shows the AWS sign-up page with the 'Contact Information' section. The page is in Korean. The 'Contact Information' section is titled '연락처 정보' and includes the following fields and options:

- 이름 (Name):** A text input field.
- 전화번호 (Phone Number):** A text input field with a dropdown menu for country code (currently set to '+1').
- 주소 (Address):** A text input field.
- 이메일 (Email):** A text input field.
- 약관 동의 (Terms and Conditions):** A checkbox labeled '약관 동의' (I agree to the terms and conditions).
- 계정 유형 (Account Type):** A dropdown menu with options: '개인' (Personal), '기업' (Business), and '교육' (Education).
- 계정 유형에 따라 추가 질문 (Additional questions based on account type):** A section with a dropdown menu for '계정 유형' (Account Type) and a text input field for '추가 질문' (Additional questions).

AWS 가입하기

다음과 같은 '결제 정보' 화면이 나타남.

계정을 생성하려면 해외 결제를 할 수 있는 신용카드 또는 체크카드가 필요함.
카드 정보를 입력하고 [확인 및 계속]을 클릭함

AWS 가입하기

7. 다음과 같이 '카드 정보 입력' 화면이 나타남.
비밀번호와 생년월일을 차례로 입력한 후, '동의'에 체크하고 [다음]을 클릭함

카드 정보 입력

카드 정보를 입력하십시오.

카드사

신용카드 | 체크카드 | 계좌입금 | 기타

이메일주소

이메일 주소를 입력하십시오.

생년월일

생년월일을 입력하십시오.

☒ 개인정보 처리에 동의합니다.

☐ 서비스 이용약관에 대한 동의

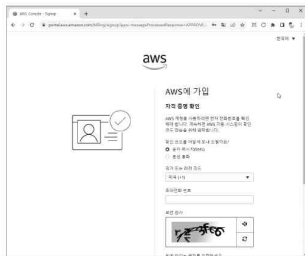
1. 선택 옵션이 아닌 필수 항목을 표시합니다.

다음

AWS 가입하기

이제 '자격 증명 확인' 화면이 나타남.

자격 증명 확인을 위해 자신의 휴대전화 번호를 입력하고 [SMS 전송]을 클릭함



AWS 가입하기

다음과 같은 '코드 확인' 화면이 나타남.

휴대전화로 전달된 4자리 확인 코드(verification code)를 입력하고 [계속]을 클릭함



AWS 가입하기

10

다음과 같은 'Support 플랜 선택' 화면이 나타남.
'기본 지원 - 무료' Support 플랜을 선택하고 [가입 완료]를 클릭함

Support 플랜 선택

비즈니스 또는 개인 계정에 대한 Support 플랜을 선택합니다. 플랜 및 요금 예시를 비교 [▶](#)해 보세요.
언제든지 AWS Management Console에서 플랜을 변경할 수 있습니다.

☒ 기본 지원 - 무료

- AWS를 처음 시작하는 신규 사용자에게 권장
- AWS 리소스에 대한 연중무휴 24시간 셀프 서비스 액세스
- 계정 및 청구 문제 전용
- Personal Health Dashboard 및 Trusted Advisor에 대한 액세스



☐ 개발자 지원 - 시작가는 29 USD/월

- AWS를 체험해보는 개발자에게 권장
- 업무 시간 중 AWS Support에 대한 이메일 액세스
- 12시간(업무 시간 기준) 이내의 응답 시간



☐ 비즈니스 지원 - 시작가는 100 USD/월

- AWS 기반 프로덕션 워크로드 실행에 추천
- 이메일, 전화 및 채팅을 통한 연중무휴 24시간 기술 지원
- 1시간 이내의 응답 시간
- Trusted Advisor 모범 사례 권장 사항 전체 세트



. AWS 가입하기

11. 그럼 다음과 같은 가입 완료 화면이 나타남



축하합니다!

AWS에 가입해 주셔서 감사합니다.

지금 계정을 활성화하고 있으며 몇 분 이내에 완료됩니다. 활성화가 완료되면 이메일을 받게 됩니다.

[AWS Management Console로 이동](#)

[다른 계정을 등록하거나 영업 팀에 문의](#)

AWS 라이트세일 시작하기

AWS 라이트세일에 접속하기

1. AWS 라이트세일 홈페이지
(<https://lightsail.aws.amazon.com>)에
접속해 보자.
홈페이지에 접속하면
다음과 같은 로그인 화면이 나타남.
루트 사용자 이메일 주소를 입력한 후
[다음]을 클릭함



로그인

☒ 루트 사용자

무제한 액세스 권한이 필요한 작업을 수행하는 계
정 소유자입니다. 자세히 알아보기

☐ IAM 사용자

일일 작업을 수행하는 계정 내 사용자입니다. 자세
히 알아보기

루트 사용자 이메일 주소

username@example.com

다음

AWS SKILL BUILDER

팀 학습의 효과 증대

필요한 리소스를
한곳에서 모두 제공하는
팀 클라우드 교육
서비스를 구독하세요.



AWS 라이트세일 시작하기

AWS 라이트세일에 접속하기

2. 이어서 비밀번호를 입력하고 [로그인]을 클릭함



AWS 라이트세일 시작하기

서버(하드웨어) 역할을 하는 라이트세일의 서비스

AWS 라이트세일에 접속하기



AWS 라이트세일 시작하기

인스턴스 생성하기

- 인스턴스를 생성하기 위해 '인스턴스 이미지 선택'에서 다음 순서로 선택해 보자



AWS 라이트세일 시작하기

인스턴스 생성하기

4. 이어서 스크롤을 내려 인스턴스명(Ubuntu-1)을 확인한 뒤, [인스턴스 생성]을 클릭해 보자



Good afternoon!

2분이상 걸림

Sort by Date ▼



Ubuntu-1

512 MB RAM, 2 vCPUs, 20 GB SSD

Pending

3.38.97.183

2406:da12:d5a:e900:ad74:9fe0:e024:ab14

Seoul, Zone A

Good afternoon!

Sort by Date ▼



Ubuntu-1

512 MB RAM, 2 vCPUs, 20 GB SSD

Running

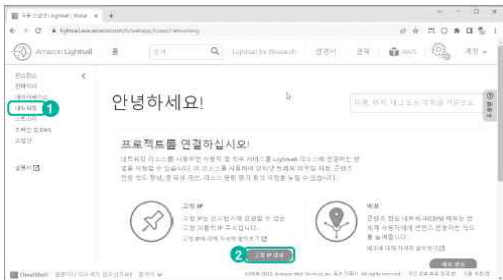
3.38.97.183

2406:da12:d5a:e900:ad74:9fe0:e024:ab14

Seoul, Zone A

고정 IP 생성하기

1. AWS 라이트세일의 메인 화면 왼쪽에 있는 [네트워킹] 탭을 선택한 후, [고정 IP 생성]을 클릭함



고정 IP 생성하기

2. 다음과 같은 화면이 등장하고, '인스턴스에 연결'에서 [Ubuntu-1]을 선택하고 고정 IP명을 확인한 후(원하는 이름으로 입력한 후), [생성]을 클릭해 고정 IP를 생성해 보자
- 인스턴스에 연결

고정 IP를 연결하면 해당 인스턴스의 동적 IP 주소가 바뀝니다.



우분투-1

512MB RAM, vCPU 2개, 20GB SSD

우분투

취소



고정 IP 식별

Lightsail 리소스에는 고유한 이름이 있어야 합니다.

miso

고정 IP 생성하기

- 다음과 같이 고정 IP 주소를 얻게 됨. 고정 IP 주소는 모두 다름



미소

고정 IP 연결됨

서울, 전 구역(ap-northeast-2)

고정 IP

세부 정보

도메인

삭제하다

공용 고정 IP 주소

이 고정 IP는 전 세계적으로 공용 연결에 사용할 수 있습니다.

43.200.87.222

인스턴스에 연결

고정 IP를 연결하면 해당 인스턴스의 동적 IP 주소가 바뀝니다.



우분투-1

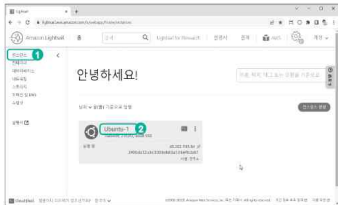
512MB RAM, vCPU 2개, 20GB SSD

우분투

폐다 ✕

방화벽 해제하기

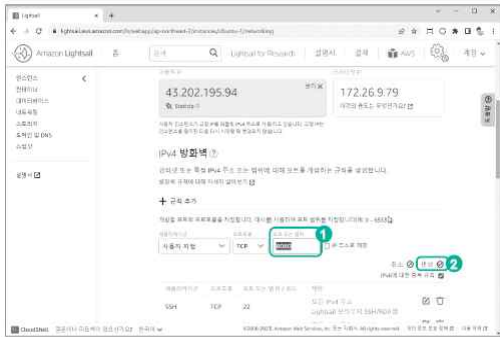
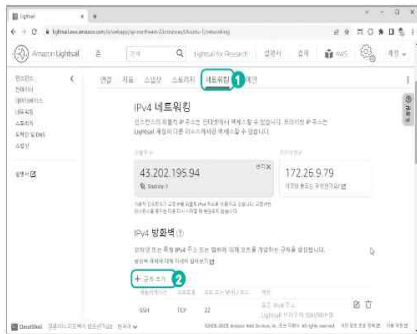
- 8080번 포트의 방화벽을 해제
- AWS 라이트세일의 메인 화면 왼쪽에 있는 [인스턴스] 탭을 선택한 후, [Ubuntu-1]을 클릭함



4-03

서버 접속 설정하기

방화벽 해제하기



· 프라이빗 키 만들기

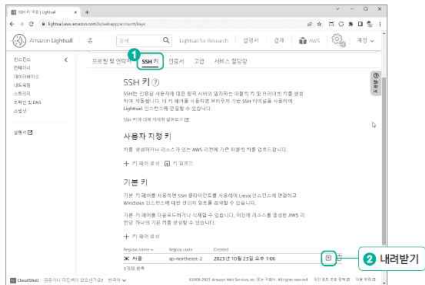
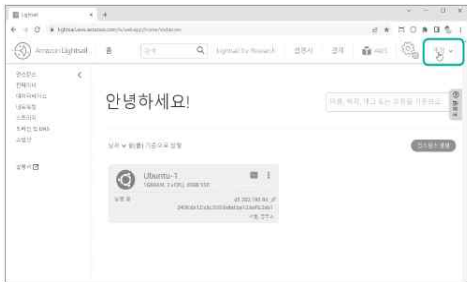
- SSH 또는 SFTP 프로그램으로 서버에 접속하기 위해서는 AWS 계정의 SSH 키가 필요함
- SSH 키는 프라이빗 키(private key)로, 서버에 안전하게 접속하기 위해 필요함
- SSH(Secure Shell)는 네트워크상의 다른 컴퓨터에 로그인하거나 원격 시스템(서버)의 명령을 수행하는 프로토콜 또는 프로그램
- SFTP(SSH File Transfer Protocol)는 SSH 프로토콜 위에서 파일을 안전하게 전송하고 관리하는 역할을 함
- 원격 서버 간에 파일을 업로드하거나 내려받을 때 SFTP를 사용함

4-04

서버 접속 프로그램 설치하기

프라이빗 키 만들기

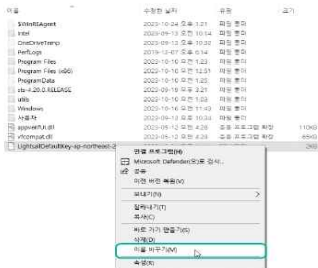
AWS 라이트세일 메인 화면에서 [계좌 → 계정]을 선택



LightsailDefaultKey-ap-northeast-2.pem라는 SSH 키를 내려받을 수 있음

프라이빗 키 만들기

- 1 내려받은 SSH 키를 컴퓨터의 루트(C:/) 디렉터리에 붙여 넣고 '이름 바꾸기'로 파일명을 mysite.pem으로 변경하자

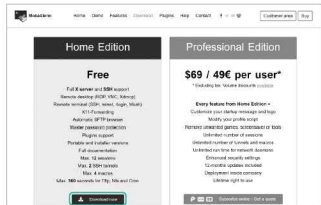


· SSH 클라이언트 설치하기

- SSH 클라이언트
 - - SSH 프로토콜을 사용하여 원격 시스템 또는
 - - 서버에 접속하여 명령을 실행할 수 있는 프로그램(터미널 프로그램)
 - - 무료로 사용할 수 있는 MobaXterm을 사용

SSH 클라이언트 설치하기

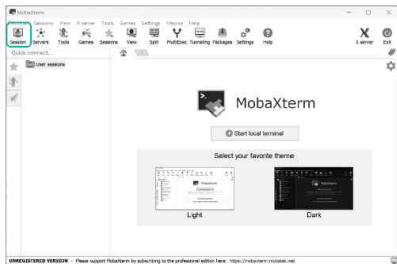
1. `mobaxterm.mo batek.net/download.html` -> 무료 버전인 Home Edition 설치파일 다운로드



[Download now]를 클릭 Installer edition을 선택해 내려받아 설치

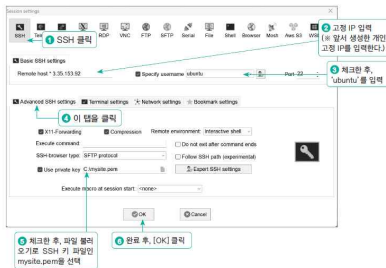
SSH 클라이언트 설치하기

2. 설치를 완료했으면 다음과 같이 MobaXterm을 실행하고 [Session]을 클릭해 보자



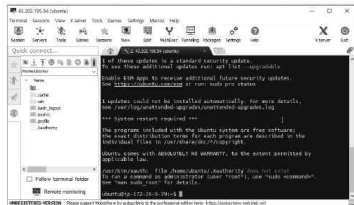
SSH 클라이언트 설치하기

- 1 Session setting 창에서 [SSH]를 클릭한 후, 다음 순서대로 내용을 입력해 보자



SSH 클라이언트 설치하기

설정을 모두 입력하면 다음과 같이 MobaXterm으로 AWS에 생성한 인스턴스 서버에 접속할 수 있음.



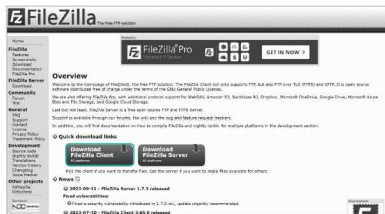
이제 MobaXterm을 이용하여 서버 작업을 할 수 있음

· SFTP 클라이언트 설치하기

- SBB 서비스는 JARJava ARchive 형태로 서버에 배포할 것임
- 즉, 배포 파일로 만든 jar 파일을 서버에 전송해야 함
- 이렇게 파일을 서버에 전송하려면 SFTP 클라이언트 프로그램도 별도로 필요함
- 여기서는 파일질라(FileZilla)를 사용함

SFTP 클라이언트 설치하기

1. <https://filezilla-project.org/> 설치 파일을 내려받아 설치



[Download FileZilla Client]를 선택하면, 다운로드 페이지에서 다시 [Download FileZillaClient] 클릭함

SFTP 클라이언트 설치하기

2. 새로 등장한 창에서 [Download]를 클릭하여 파일질라를 내려받음



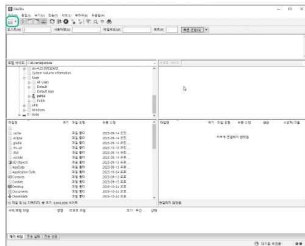
4-04

서버 접속 프로그램 설치하기

SFTP 클라이언트 설치하기

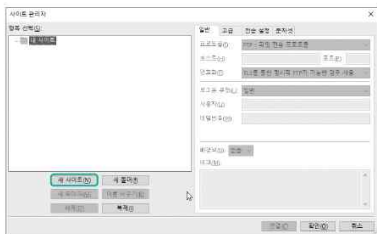
설치를 완료했으면 파일질라를 실행해 보자.

그다음, 화면 왼쪽 상단의 [사이트 관리자 열기] 아이콘을 클릭해 보자



SFTP 클라이언트 설치하기

아이콘을 클릭하면 오른쪽과 같이 사이트 관리자 화면이 나타남.
그다음, [새 사이트] 버튼을 클릭함



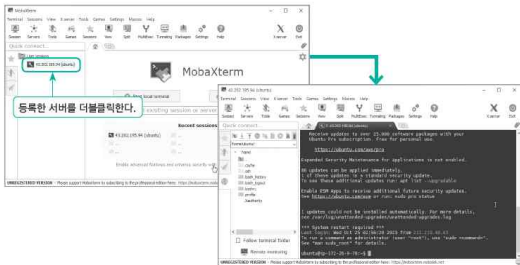
SFTP 클라이언트 설치하기

사이트 관리자 창에서 다음 순서대로 내용을 입력해 보자



서버 환경 설정하기

먼저 터미널(mobaXterm)을 사용하여 서버에 접속해 보자.
왼쪽 화면의 메뉴에서 등록한 서버를 더블클릭하면 오른쪽과 같은 터미널 창이 등장함



서버 환경 설정하기

호스트 이름 변경하기

```
ubuntu@ip-172-26-1-61:~$
```

1. 터미널 창에 다음과 같이 'jumpto'로 호스트 이름을 바꿔 보자

```
ubuntu@ip-172-26-1-61:~$ sudo hostnamectl set-hostname jumpto
```

1. 호스트명을 바꾼 후에는 서버를 다시 시작해야 적용됨

```
ubuntu@ip-172-26-1-61:~$ sudo reboot
```

서버 환경 설정하기

호스트 이름 변경하기

1. `sudo reboot` 명령을 수행하면 서버가 재시작되므로 연결된 접속이 끊어짐.
잠시 후에 다시 접속하면 다음과 같이 변경된 프롬프트를 확인할 수 있음

```
ubuntu@jumpo:~$
```

1. `hostname` 명령을 수행하면 다음과 같이 변경된 호스트명이 출력됨

```
ubuntu@jumpo:~$ hostname  
jumpo
```

서버 환경 설정하기

서버 시간 설정하기

서버 시간을 설정하기 위해 터미널에서 `date` 명령을 실행해 보자.
그러면 다음과 같이 우리나라 시간이 아닌 UTC 시간이 출력될 것임.

```
ubuntu@jumpo:~$ date  
Mon Mar 21 03:32:36 UTC 2022
```

UTC 시간은 국제 표준 시간이므로, SBB에서 게시물의 등록 시간을
우리나라 시간으로 맞추려면 서버 시간 설정을 바꿔야 함

서버 환경 설정하기

서버 시간 설정하기

2. 한국 시간(KST)으로 설정하기 위해 다음 명령을 입력해 보자

```
ubuntu@jumpo:~$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
```

1. 다시 date 명령을 수행하면 우리나라 시간으로 출력됨

```
ubuntu@jumpo:~$ date  
Mon Mar 21 12:33:30 KST 2022
```

서버 환경 설정하기

서버에 자바 설치하기

서버에 자바가 설치되어 있는지를 확인하기 위해 다음과 같이 java 명령을 입력해 보자

```
ubuntu@jumpo:~$ java
```

```
Command 'java' not found, but can be installed with:
```

자바가 설치되지 않았다는
메시지를 출력한다.

```
sudo apt install openjdk-11-jre-headless # version 11.0.20.1+1-0ubuntu1~22.04, or
```

```
sudo apt install default-jre # version 2:1.11-72build2
```

```
sudo apt install openjdk-17-jre-headless # version 17.0.8.1+1-us1-0ubuntu1~22.04
```

```
sudo apt install openjdk-18-jre-headless # version 18.0.2+9-2~22.04
```

```
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-0ubuntu3~22.04
```

```
sudo apt install openjdk-8-jre-headless # version 8u382-ga-1~22.04.1
```

서버 환경 설정하기

서버에 자바 설치하기

2. 자바를 설치하기 전에 먼저 `sudo apt update` 명령을 수행하여 우분투 패키지를 최신으로 업그레이드해야 함

SW 패키지를 설치, 업데이트, 제거 등 관리를 위한 명령어이다.

```
ubuntu@jumpo:~$ sudo apt update
Hit:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
(... 생략 ...)
```

· 서버 환경 설정하기

· 서버에 자바 설치하기

- 이어서 다음과 같이 자바를 설치해 보자

```
ubuntu@jumpo:~$ sudo apt install openjdk-19 -jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
(... 생략 ...)
```

서버 환경 설정하기

서버에 자바 설치하기

설치를 완료함. 다음과 같이 `java -version`을 실행해 보자.

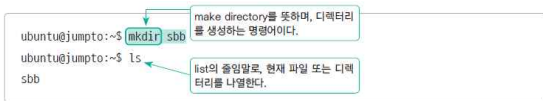
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@miso:~$ java -version  
openjdk version "17.0.11" 2024-04-16  
OpenJDK Runtime Environment (build 17.0.11+9-Ubuntu-122.04.1)  
OpenJDK 64-Bit Server VM (build 17.0.11+9-Ubuntu-122.04.1, mixed mode, sharing)  
ubuntu@miso:~$
```

이를 통해 자바 17 버전이 정상적으로 설치된 것을 확인할 수 있음

서버 환경 설정하기

프로젝트 디렉터리 생성하기

서버에 SBB 서비스를 적용하기 위해
다음과 같이 홈 디렉터리(/home/ubuntu) 하위에 sbb라는 디렉터를 생성해야 함.
mkdir sbb라는 명령어를 입력해 보자



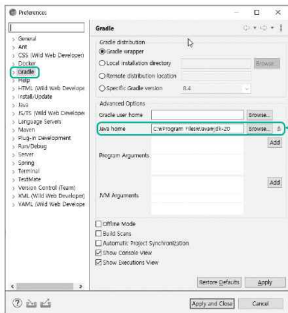
STS에서 SBB 배포 파일 생성하기

이제 서버에 적용할 SBB 배포 파일을 만들어 보자.
SBB 배포 파일은 단 하나의 jar 파일로, 서버에서는 이 jar 파일을 실행하여 SBB 서비스를 구동함

jar 파일은 Gradle을 사용하여 생성함.
jar 파일을 생성하기 전에 Gradle에 JDK 설정을 먼저 해야 함.
그러기 위해서 다음과 같이 [Window → Preferences → Gradle]를 선택함.
그러면 다음과 같은 화면이 나타남.

'Advanced Options'의 'Java home' 항목에
이와 같이 설치한 JDK의 디렉터리(예: C:\Program Files\Java\jdk-21)를 입력함

STS에서 SBB 배포 파일 생성하기



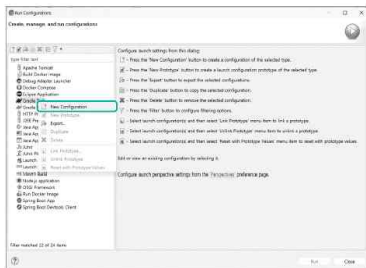
설치한 JDK의 디렉터리
위치를 입력한다.

STS에서 SBB 배포 파일 생성하기

서버에 적용할 SBB 배포 파일(jar 파일)을 만들기 위해 다시 STS로 돌아가 메뉴에서 [Run → Run Configurations]을 선택해 보자.

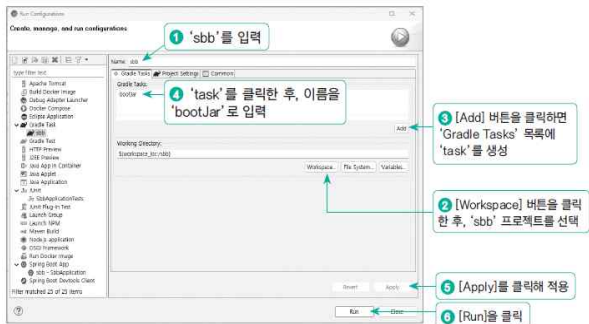
그러면 다음 화면이 나타남.

'Gradle Task'를 선택하고
마우스 오른쪽 버튼을 눌러
[NewConfiguration]을 클릭하자



STS에서 SBB 배포 파일 생성하기

다음과 같은 화면이 나타나면 순서대로 각 항목에 다음과 같이 입력해 보자.



여기까지 진행하면 왼쪽에 있는 'Gradle Task'에 'sbb' 항목이 추가됨.
항목을 확인한 후, 화면에서 [Run] 버튼을 눌러 배포 파일을 만들어 보자

STS에서 SBB 배포 파일 생성하기

- 배포 파일을 완성했다면 STS 화면 하단에 다음과 같이 Gradle Executions 창이 나타나고 배포 파일이 생성됨을 표시하는 진행 결과가 나타날 것임



The screenshot shows the 'Gradle Executions' window in the STS IDE. It displays a tree view of build operations and their durations. The operations are grouped into 'Run build' and 'Run main tasks'.

Operation	Duration
Run build	21.893 s
Load build	0.034 s
Configure build	2.352 s
Calculate build tree task graph	0.249 s
Build phase: Configure build	2.352 s
Build finished for file system watching	0.000 s
Run main tasks	19.229 s
Run tasks	19.226 s
Build phase: Run tasks	19.226 s
Build started for file system watching	0.002 s
Build phase: Run main tasks	19.229 s

STS에서 SBB 배포 파일 생성하기

여기까지 오류 없이 잘 실행되었다면
SBB 프로젝트 디렉터리의 하위에 build 디렉터리가 생기고
다음과 같은 파일이 생성된 것을 확인할 수 있음



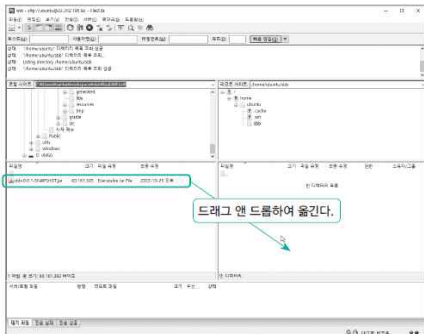
STS에서 SBB 배포 파일 생성하기

이때 생성되는 sbb-0.0.2.jar라는 파일 이름은
프로젝트명 sbb와 build.gradle 파일의 **version** 항목값을 조합해 만들

```
4      id 'io.spring.dependency-management'
5  }
6
7  group = 'com.mysite'
8  version = '0.0.2'
9
10 }
```


SFTP로 SBB 배포 파일 전송하기

다음과 같이 sbb-0.0.2.jar
파일을 찾아 서버의
/home/ubuntu/sbb 디렉터리로
드래그 앤 드롭해 보자



SFTP로 SBB 배포 파일 전송하기

터미널 프로그램인 MobaXterm을 사용하여 다시 서버에 접속한 후, 다음과 같이 입력하여 배포 파일을 실행해 보자

```
ubuntu@jumpo:~$ cd sbb
ubuntu@jumpo:~/sbb$ java -jar sbb-0.0.1-SNAPSHOT.jar
```

이와 같이 입력하여 SBB 서비스를 실행한다.

```

      _   _       _   _       _   _ 
     / \   \     / \   \     / \   \
    ( )   | |   | |   | |   | |   | |
   / \   | |   | |   | |   | |   | |
  / \   | |   | |   | |   | |   | |
   ^   | |   | |   | |   | |   | |
 =====|_|=====|_|_/_/_/_/
:: Spring Boot ::                (v3.1.5)

```

```

2023-10-25T04:25:25.091Z INFO 5685 --- [          main] com.mysite.sbb.SbbApplication : Starting SbbApplication v0.0.1-SNAPSHOT using Java 19.0.2
with PID 5685 (/home/ubuntu/sbb/sbb-0.0.1-SNAPSHOT.jar started by ubuntu in /home/ubuntu/sbb)
2023-10-25T04:25:25.095Z INFO 5685 --- [          main] com.mysite.sbb.SbbApplication : No active profile set, falling back to 1 default profile: "default"
(... 생략 ...)
```

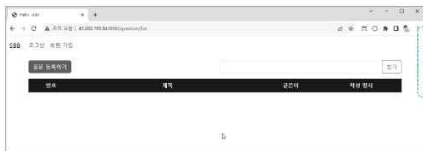
SFTP로 SBB 배포 파일 전송하기

'change directory'를 의미하는 cd 명령어를 사용하여 sbb-0.0.2.jar 파일이 있는 디렉터리로 이동한 후, 다음 명령을 입력하면 SBB 서비스가 실행됨

```
java -jar sbb-0.0.1-SNAPSHOT.jar
```

SFTP로 SBB 배포 파일 전송하기

웹 브라우저를 실행하고 앞서 설정한 고정 IP에 포트 번호 8080을 붙여 접속해 보자.
예를 들어, `http://43.202.195.94:8080`과 같이 입력하면 다음 화면을 볼 수 있음.



여기까지 따라왔다면
여러분이 만든 웹 서비스가
세상에 공개된 것이야!



이제 웹 브라우저에서 `http://43.202.195.94:8080`와 같은 형식의 URL을 입력하면
누구나 이 서비스를 사용할 수 있음

4-05절에서 다음 명령을 입력하여 SBB 서비스를 실행함

```
ubuntu@jumpo:~/sbb$ java -jar sbb-0.0.1-SNAPSHOT.jar
```

하지만 이같이 실행하면 서버에 접속한 터미널 프로그램을 종료할 경우 SBB 서비스도 중단됨

따라서 이를 방지하기 위해서는 백그라운드로 서비스를 실행해야 함

백그라운드(background)란 실행되는 프로그램이나 프로세스가 현재 프로세스와 별개로 실행되는 것을 의미함

이번에는 백그라운드로 SBB 서비스를 시작하는 스크립트와 서비스를 중지하는 스크립트를 만들어 보자

SBB 시작 스크립트 작성하기

- 백그라운드에서 SBB 서비스를 실행하려면 시작 스크립트를 작성해야 함
- 이때 서버에서는 STS와 같은 IDE를 사용할 수 없으므로 나노(nano) 편집기를 사용해야 함
- 나노 편집기란 리눅스 기반 OS에서 사용할 수 있는 간단한 터미널 텍스트 편집기임

· MobaXterm에서 sbb 디렉터리로 이동한 다음, nano start.sh 명령을 입력해 보자

```
ubuntu@jumpo:~$ cd sbb ← 이미 sbb 디렉터리로 이동했다면 이 명령은 생략한다.  
ubuntu@jumpo:~/sbb$ nano start.sh
```

여기서 nano start.sh는 나노 편집기를 통해 start.sh라는 스크립트 파일을 생성 또는 편집하라는 명령문임

SBB 시작 스크립트 작성하기

2. 명령을 실행하면 다음과 같은 화면이 나타남. 이 화면이 바로 나노 편집기임.
아쉽게도 나노 편집기에서는 마우스를 사용할 수 없음.
하지만 화살표 키로 커서를 움직이며 편집할 수 있으므로 그다지 어렵지 않음.
편집기 아래에는 여러 단축키 기능이 표시되어 있음



SBB 시작 스크립트 작성하기

나노 편집기에서 다음 내용을 입력해 보자.

```
#!/bin/bash

JAR=sbb-0.0.1-SNAPSHOT.jar
LOG=/home/ubuntu/sbb/sbb.log

nohup java -jar $JAR > $LOG 2>&1 &
```

• /home/ubuntu/sbb/start.sh

배포 파일 이름을 입력한다.

로그를 출력할 서버 로그 파일 이름을 입력한다.

이때 JAR 변수는 배포 파일 이름이고, LOG 변수는 로그를 출력할 파일 이름임.
start.sh에서 nohup는 프로세스를 실행한 터미널의 연결이 끊어지더라도
프로세스가 지속적으로 동작할 수 있게 해주는 명령어이고,
이어서 java -jar \$JAR는 JAR 변수에 저장된 JAR 파일을 실행하라는 명령어임

SBB 시작 스크립트 작성하기

명령어 `> $LOG`는 자바로 실행된 프로세스의 출력을 로그 파일에 저장하라는 의미임.
`2>&1`은 오류 출력(stderr)을 일반 출력(stdout)으로 전달하라는 의미임.

따라서 일반 로그와 오류 로그가 모두 `sbb.log` 파일에 저장될 것임.

마지막의 `&` 기호는 백그라운드로 명령을 실행하라는 의미임.

`Ctrl + O`를 누른 후, 화면 아래에 'File Name to Write: '이라는 프롬프트가 등장하면 `Enter`를 눌러 `start.sh` 파일을 저장함.

그다음 `Ctrl + X`를 눌러서 편집기를 종료하자

SBB 시작 스크립트 작성하기

편집기를 빠져나와 다음과 같이 start.sh이라는 스크립트명만 입력하더라도 실행될 수 있도록 실행 권한을 부여하자

```
ubuntu@jumpo:~/sbb$ chmod +x start.sh
```

'change mode'의 줄임말로, 파일의 권한 모드를 변경하라는 뜻이다.

+는 권한을 추가한다는 뜻이고, x는 실행할 수 있는 권한을 뜻한다.

SBB 시작 스크립트 작성하기

- 다음과 같이 start.sh 파일을 실행하여 백그라운드로 서버를 실행해 보자

```
ubuntu@jumpo:~/sbb$ ./start.sh  
ubuntu@jumpo:~/sbb$
```

./는 현재 디렉터리를
뜻한다.

SBB 중지 스크립트 작성하기

start.sh를 만들었을 때와 마찬가지로 nano 편집기를 통해 stop.sh 파일을 생성한 후, stop.sh 스크립트를 다음과 같이 작성해 보자

```
• /home/ubuntu/sbb/stop.sh

#!/bin/bash

SBB_PID=$(ps -ef | grep java | grep sbb | awk '{print $2}')

if [ -z "$SBB_PID" ];
then
    echo "SBB is not running"
else
    kill -9 $SBB_PID
    echo "SBB stopped"
fi
```

SBB 중지 스크립트 작성하기

SBB_PID는 현재 실행 중인 SBB 서비스의 프로세스 아이디임.
프로세스 아이디는 이와 같이 ps 명령어와 grep 그리고 awk 명령어를 조합하여 찾을 수 있음.

조금 더 자세히 알아보면 ps -ef 명령은 현재 실행 중인 프로세스를 출력하고
grep java | grep sbb 명령은 출력된 문자열에서 java와 sbb라는 문자열이 포함된
프로세스만 필터링하여 출력함.

그리고 awk '{print \$2}' 명령은 출력 문자열의 2번째 항목인
프로세스 아이디만 뽑아내는 역할을 함.

이러한 과정을 통해 SBB_PID값을 구할 수 있음.
만약 SBB 서비스의 프로세스가 없다면 'SBB is not running'이라는 메시지를 출력하고,
프로세스가 있다면 kill -9이라는 명령문으로 해당 프로세스를 강제로 종료함

SBB 중지 스크립트 작성하기

- 2 편집기를 빠져나와 다음과 같이 스크립트명만 입력하더라도 실행될 수 있게 실행 권한을 부여하자

```
ubuntu@jumpo:~/sbb$ chmod +x stop.sh
```

- 1 다음과 같이 stop.sh 파일을 실행하여 SBB 서비스를 중지해 보자

```
ubuntu@jumpo:~/sbb$ ./stop.sh
```

SBB 중지 스크립트 작성하기

만약 SBB 프로그램이 변경되어 jar 파일을 새로 업로드했다면
변경된 내용을 적용하기 위해 stop.sh와 start.sh를 순서대로 실행하면 됨

```
ubuntu@jumpo:~/sbb$ ./stop.sh  
ubuntu@jumpo:~/sbb$ ./start.sh
```

- SBB 서비스는 DB로 H2 데이터베이스를 사용해 왔음
- 그동안 좀 더 편리하게 개발하기 위해
H2 데이터베이스의 사용자명은 sa, 비밀번호는 빈값("")으로 설정했지만
실제 운영 환경, 특히 AWS와 같은 클라우드 서버에서는
이와 같은 설정은 보안상 매우 위험함
- 기본 사용자명과 비밀번호를 그대로 사용하면
누구나 H2 데이터베이스 콘솔에 로그인할 수 있기 때문임
- 따라서 서버 환경과 개발 환경을 분리하고, 서버 환경에서는
보안을 위해 DB에 비밀번호를 설정하는 것이 필요함
- 이번에는 개발 환경과 서버 환경을 분리하고,
서버 환경에서 H2 데이터베이스의 비밀번호를 설정하는 방법을 알아보자

서버 환경 파일 생성하기

- H2 데이터베이스에 비밀번호를 설정하는 방법은 매우 간단함
- STS에서 src/main/resource 디렉터리의 application.properties 파일을 열어 '1234'와 같이 비밀번호를 입력하기만 하면 됨.

```
spring.datasource.password=1234
```

- 하지만 개발 편의를 위해 로컬 환경은 여전히 비밀번호 없이 사용하고 서버 환경에만 비밀번호를 설정하고 싶다면 어떻게 하면 될까?

서버 환경 파일 생성하기

스프링 부트는 시작 옵션으로 다음과 같이 `spring.profiles.active` 항목을 전달할 수 있음

```
java -Dspring.profiles.active=prod -jar sbb-0.0.1-SNAPSHOT.jar
```

이처럼 `-Dspring.profiles.active=prod` 옵션을 주어 실행하면
스프링 부트는 `application.properties` 파일 대신 `application-prod.properties`를 사용함

만약 `-Dspring.profiles.active=abc` 옵션을 주어 실행하면
서버 환경 파일로 `application-abc.properties` 파일을 사용하게 됨

· 서버 환경 파일 생성하기

- STS에서 `src/main/resources` 디렉터리에 `application.properties` 파일을 클릭해 마우스 오른쪽 버튼을 누른 후 [Copy → Paste]하여 `application-prod.properties` 파일을 생성해 보자
- 그다음 서버에서 H2 데이터베이스를 사용하기 위해 `application-prod.properties` 파일을 다음과 같이 작성해 보자

서버 환경 파일 생성하기

```
• application-prod.properties

# DATABASE
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.h2.console.settings.web-allow-others=true
spring.datasource.url=jdbc:h2:~/local
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=1234

# JPA
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true
```

application.properties와 달리
여기서는
spring.datasource.password에 비밀
번호를 설정하고

외부에서도 H2 콘솔에
접근할 수 있도록
spring.h2.console.settings.web-
allowothers=true로 설정함

빌드 버전 변경하여 배포 파일 생성하기

- 환경 파일을 생성했으므로 SBB가 업데이트됨
- build.gradle 파일의 빌드 버전도 다음과 같이 변경해 주자
- 빌드 버전을 바꾸지 않으면 이전 배포 파일과 동일한 jar 파일명으로 생성되므로 새로운 배포 파일을 생성할 때는 빌드 버전을 변경하는 것이 좋음

```
• build.gradle

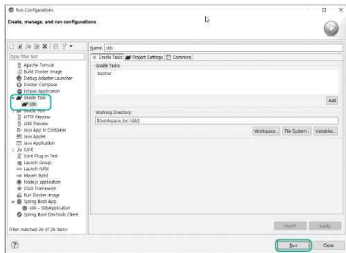
(... 생략 ...)

group = 'com.mysite'
version = '0.0.2'

(... 생략 ...)
```

빌드 버전 변경하여 배포 파일 생성하기

이와 같이 수정한 후 다음과 같이 Run Configuration 창에서 Gradle Task의 sbb를 선택하고 [Run] 버튼을 클릭하여 배포 파일을 새로 만들자



· 빌드 버전 변경하여 배포 파일 생성하기

- 그러면 다음과 같은 배포 파일이 생성됨

```
workspace/sbb/build/libs/sbb-0.0.2.jar
```

- 배포 파일을 생성했으니 sbb-0.0.2.jar 파일도
파일질라를 사용하여 서버의 sbb 디렉터리에 업로드하자

서버에 변경 내용 적용하기

데이터베이스에 비밀번호를 설정했기 때문에
기존에 있던 데이터베이스 파일을 지워야 함.

왜냐하면 기존에 생성된 데이터베이스의 비밀번호와
현재 접속하려는 데이터베이스의 비밀번호가 다르기 때문임.

먼저, MobaXterm으로 돌아가 다음과 같이 차례로 입력하여
기존의 데이터베이스 파일을 삭제하자

```
ubuntu@jumpo:~$ ls
```

```
local.mv.db sbb
```

```
ubuntu@jumpo:~$ rm local.mv.db
```

remove를 뜻하는 리눅스 명령어이다.

서버에 변경 내용 적용하기

- 다시 sbb 디렉터리로 이동한 후,
나노 편집기에서 서버의 start.sh 파일도 다음과 같이 수정하자.

```
• /home/ubuntu/sbb/start.sh

#!/bin/bash

JAR=sbb-0.0.2.jar
LOG=/home/ubuntu/sbb/sbb.log

nohup java -Dspring.profiles.active=prod -jar $JAR > $LOG 2>&1 &
```

새로 업로드한 sbb-0.0.2.jar를 적용하고
-Dspring.profiles.active=prod 옵션을 주어 스프링 부트가 실행되도록 설정함

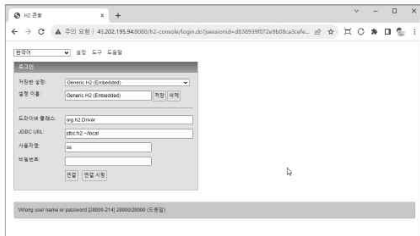
· 서버에 변경 내용 적용하기

수정을 완료하면 다음과 같이 입력하여 SBB 서비스를 재시작하자

```
ubuntu@jumpo:~/sbb$ ./stop.sh  
SBB stopped.  
ubuntu@jumpo:~/sbb$ ./start.sh
```

서버의 H2 콘솔 접속하기

서버의 H2 콘솔에 접속하여 다음과 같이 비밀번호 없이 접속을 시도해 보자.

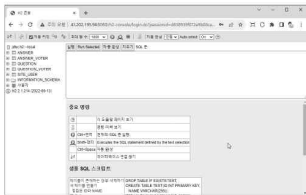


이와 같이 비밀번호가 잘못되었다는 오류 메시지를 확인할 수 있을 것임

서버 환경으로 분리하기

· 서버의 H2 콘솔 접속하기

- 다시 서버 환경 파일에 설정한 비밀번호를 입력하여 로그인을 다시 시도해 보자.
로그인이 잘 수행되어 우리가 만든 테이블을 확인할 수 있을 것임.



개발 환경과 서버 환경을 다르게 설정해야 할 때
이와 같이 환경 파일을 분리하여 사용할 수 있음

- 그동안 개발 단계에서 웹 프로그램을 개발하고 테스트하기 위해 로컬 서버의 8080번 포트를 사용함
- 하지만 운영 환경에서는 실제 가장 많이 사용하는 80번 포트를 사용해야 함
- 다음과 같이 -Dserver.port=80와 같은 옵션을 추가하여 스프링 부트 서비스를 실행하면 80번 포트로 서비스를 운영할 수 있음
- 즉, 다음과 같이 실행하면 80번 포트로 서비스할 수 있음

```
$ sudo java -Dserver.port=80 -jar sbb-0.0.2.jar
```

- 하지만 이와 같이 80번 포트를 직접 지정해 사용하는 방법은 추천하지 않음
- 80번 포트로 서버를 실행하려면 루트 권한이 필요하다는 단점도 있고, SSL을 적용한 HTTPS 서비스를 운영하기가 쉽지 않기 때문
- 이러한 이유로 엔진엑스(Nginx)나 아파치(Apache)와 같은 웹 서버를 사용해 80번 포트로 웹서비스를 제공하려고 함
- 그중에서도 엔진엑스는 높은 성능을 위해서 개발된 웹 서버로, 설정이 간단하고 쉽게 사용할 수 있음
- 이번에는 엔진엑스와 스프링 부트를 연동하여 80번 포트로 우리가 만든 SBB 서비스를 운영하는 방법을 알아보자

엔진엑스 설치 및 설정하기

MobaXterm에서 다음과 같이 입력하여 루트 권한으로 엔진엑스를 설치해 보자.

```
ubuntu@jumpo:~$ sudo apt install nginx
```

엔진엑스 설치 시 등장하는 질문에는
'Yes'를 선택한 후,
다음 화면에서는 [OK]를 선택하자



엔진엑스 설치 및 설정하기

- 설치를 완료한 후, /etc/nginx/sites-available 디렉터리로 이동해 보자.

```
ubuntu@jumpo:~$ cd /etc/nginx/sites-available
```

/etc/nginx/sites-available 디렉터리는 엔진엑스의 설정 파일이 위치한 디렉터리임.
처음 설치했을 때는 이 디렉터리 안에 'default'라는 설정 파일만 존재함

엔진엑스 설치 및 설정하기

SBB 서비스의 엔진엑스 설정 파일을 다음과 같이 루트 권한으로 생성해 보자.

```
ubuntu@jumpo:/etc/nginx/sites-available$ sudo nano sbb
```

나노 편집기에서 sbb 파일의 내용을 다음과 같이 작성해 보자

엔진엑스 설치 및 설정하기

1

• /etc/nginx/sites-available/sbb

```
server {  
    listen 80;  
    server_name localhost;  
  
    location / {  
        proxy_pass http://localhost:8080;  
        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;  
    }  
}
```

이 코드 블록 안에 웹 서버
의 설정을 정의한다.

엔진엑스 설치 및 설정하기

작성한 내용을 살펴보면 listen 80은 웹 서버를 80번 포트로 서비스한다는 의미임.

이제 `http://43.202.195.94:8080/`와 같은 URL에서 포트 번호를 생략하고 `http://43.202.195.94`와 같이 입력하여 웹 브라우저에서 접속할 수 있음.

아직은 ip와 연결된 도메인을 구입하지 않았으므로 `server_name`에는 `localhost`를 입력함

엔진엑스 설치 및 설정하기

1. location / { ... }은 /로 시작되는 모든 URL,
즉 모든 클라이언트 요청에 대한 설정을 담당하는 영역임.
여기서 작성한 세부 설정의 내용은 다음과 같음

- proxy_pass: 엔진엑스 웹 서버가 받은 모든 클라이언트 요청을 http://localhost:8080으로 리다이렉트한다.
- proxy_set_header: 브라우저에서 SBB 서비스를 호출하면 엔진엑스를 통해서 스프링 부트의 톰캣 서버로 요청이 전달된다. proxy_set_header 설정은 이 과정에서 클라이언트의 주소가 실제 IP 주소가 아닌 엔진엑스가 설치된 서버의 주소로 톰캣 서버에 전달되는 것을 방지하기 위해 사용한다.

엔진엑스 설치 및 설정하기

이제 작성한 sbb 파일을 엔진엑스가 SBB 서비스의 설정 파일로 읽을 수 있도록 sbb 파일을 /etc/nginx/sites-enabled 디렉터리에 링크해야 함.

먼저 파일을 저장한 후, 나노 편집기를 빠져나와 site-enabled 디렉터리로 이동해 보자.

```
ubuntu@jumpo:~/etc/nginx/sites-available$ cd /etc/nginx/sites-enabled/  
ubuntu@jumpo:~/etc/nginx/sites-enabled$
```

sites-enabled 디렉터리는 site-available 디렉터리에 있는 설정 파일 중에서 활성화하고 싶은 것을 링크로 관리하는 디렉터리임

엔진엑스 설치 및 설정하기

ls 명령을 수행하면 현재 default 설정 파일만 링크되어 있음을 확인할 수 있음

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ ls  
default
```

site-enabled 디렉터리의 default 링크는 다음과 같이 삭제하자

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ sudo rm default
```

엔진엑스 설치 및 설정하기

그리고 다음과 같이 /etc/nginx/sites-available/sbb 파일을 /etc/nginx/sites-enabled/sbb 파일로 링크하자

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ sudo ln -s /etc/nginx/sites-available/sbb
```

다시 ls 명령을 수행하면 default는 사라지고 sbb 링크만 남는 것을 확인할 수 있음

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ ls  
sbb
```

엔진엑스 실행 및 적용하기

엔진엑스를 설치할 때 엔진엑스가 자동으로 실행됐으므로 엔진엑스 설정을 적용하려면 다음과 같이 입력하여 엔진엑스를 다시 시작해야 함

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ sudo systemctl restart nginx
```

엔진엑스 웹 서버를 적용했으니 웹 브라우저에서 다음과 같은 URL로 접속해 보자.
이때 각자의 고정 IP를 사용하도록 하자

```
http://43.202.195.94
```


엔진엑스 실행 및 적용하기

2. 웹 서버(엔진엑스)를 사용하기 때문에 이전과 달리 :8080과 같은 포트 번호를 사용할 필요가 없으며 HTTP 기본 포트인 80 포트를 사용할 수 있음.

80번 포트는 `http://43.202.195.94:80`과 같이 사용해도 되지만 :80을 생략할 수 있음.
다음처럼 완벽하게 실행되는 SBB를 확인할 수 있음



- 로그(log)란 프로그램 또는 시스템에서 발생하는 이벤트, 정보, 상태, 오류 등을 기록한 것을 말하고, 이러한 로그를 생성하고 저장하는 것을 로깅(logging)이라 함
- 로그는 기록된 데이터인 반면, 로깅은 생성하고 저장하는 프로세스와 도구를 모두 아우르는 말임
- 로그는 프로그램의 동작을 추적해 문제를 해결하거나 성능을 분석하기 위한 목적으로 활용됨
- 이 때문에 로그 관리는 프로그램 개발과 운영에서 아주 중요한 역할을 함

- 스프링 부트는 기본적으로 로그백(logback)이라는 로깅 도구를 사용하여 로그를 관리함
- STS의 콘솔 창에 출력되는 내용과 서버에서 sbb.log 파일에 출력되는 내용 모두가 로그백에 의해 출력되는 로그임
- 그런데 현재 서버에 생성되는 로그 파일(sbb.log)에는 다음과 같은 몇 가지 문제가 있음
 - SBB 서비스를 다시 실행할 경우(즉, stop.sh를 하고 start.sh를 실행할 경우) 이전 로그가 삭제된다.
 - 로그가 쌓일수록 로그 파일의 용량이 커지며 무한대로 증가할 수 있다.
 - 로그 시간이 시스템 시간이 아닌 UTC 시간으로 출력된다.
- 이 문제를 어떻게 해야 해결할 수 있을까?

서버 로그 분리하기

앞서 말한 서버 로그의 문제점을 해결하려면
개발 환경의 로그와 서버 환경의 로그를 분리해서 관리해야 함

그러기 위해 4-07절에서
서버 환경으로 분리한 것처럼

application-prod.properties
파일에 로깅 설정을 추가해
서버 로그를 분리해 보자

```
• application-prod.properties

# DATABASE
(... 생략 ...)

# JPA
(... 생략 ...)

# logging
logging.logback.rollingpolicy.max-history=30
logging.logback.rollingpolicy.max-file-size=100MB
logging.file.name=logs/sbb.log
logging.logback.rollingpolicy.file-name-pattern=${LOG_FILE}-%d{yyyy-MM-dd}-%i.log
logging.pattern.dateformat=yyyy-MM-dd HH:mm:ss.SSS,Asia/Seoul
```

서버 로그 분리하기

application-prod.properties 파일에
이와 같이 logging 설정을 추가함.

각각의 항목을 알아보자.

- logging.logback.rollingpolicy.max-history: 로그 파일을 유지할 기간(일수)을 설정한다. 여기서는 30일 간의 로그만 유지하도록 했다.
- logging.logback.rollingpolicy.max-file-size: 로그 파일 1개의 최대 용량(size)를 설정한다. 여기서는 100MB로 설정했다.
- logging.file.name: 로그 파일의 이름을 설정한다. 여기서 logs/sbb.log는 logs 디렉터리의 하위에 sbb.log라는 이름으로 로그 파일을 생성하라는 의미이다.
- logging.logback.rollingpolicy.file-name-pattern: 로그 파일의 용량이 설정한 용량을 초과하거나 날짜가 변경될 경우 새로이 만들어질 로그 파일의 이름에 관한 규칙(pattern)을 설정한다.
- logging.pattern.dateformat: 로그 출력 시 출력하는 날짜와 시간의 형식과 타임존^{time zone}을 설정한다. 여기서는 Asia/Seoul로 설정했다. 만약 타임존을 설정하지 않을 경우 UTC 시간을 기준으로 출력한다.

이와 같이 수정하고 빌드 버전을 변경하여 배포 파일(sbb-0.0.3.jar)을
새로 작성하여 서버에 업로드하자

서버에서 start.sh 변경하기

로깅 설정을 변경했으므로 서버에 변경 내용을 적용하기 위해 mobaXterm으로 돌아가 start.sh 파일도 다음과 같이 수정해 보자.

```
• /home/ubuntu/sbb/start.sh

#!/bin/bash

JAR=sbb-0.0.3.jar
LOG=/dev/null
export spring_profiles_active=prod

nohup java -jar $JAR > $LOG 2>&1 &
```

start.sh 실행 시 생성되는 로그 파일을 sbb.log에서 /dev/null로 변경함.
왜냐하면 앞서 로깅 설정을 통해 logs 디렉터리 하위에 로그 파일(sbb.log)이 생성되도록 설정했기 때문임. 자바 프로그램의 출력을 /dev/null로 지정하면 콘솔 출력이 무시됨

서버에서 start.sh 변경하기

2. 기존의 로그는 더 이상 필요하지 않으므로 다음과 같이 삭제하자

```
ubuntu@jumpo:~/sbb$ rm sbb.log
```

1. SBB 서비스를 다음과 같이 재시작하자

```
ubuntu@jumpo:~/sbb$ ./ stop.sh  
SBB stopped  
ubuntu@jumpo:~/sbb$ ./start.sh
```

서버에서 start.sh 변경하기

재시작하면 다음과 같이 logs 디렉터리가 생성되고,
logs 디렉터리 밑에 sbb.log 파일이 생성되는 것을 확인할 수 있음.

```
ubuntu@jumpo:~/sbb$ cd logs
ubuntu@jumpo:~/sbb/logs$ ls
sbb.log
```

이제 기존 로그도 사라지지 않고 30일 간 날짜별(용량별)로 유지되며,
우리나라 시각을 기준으로 잘 출력될 것임

· 사용자 로그 작성하기

- 지금까지는 스프링 부트 프레임워크가 출력하는 로그들을 관리하는 방법을 알아봄
- 한편 사용자, 즉 개발자가 코드를 직접 작성하여 로그를 출력할 수도 있음
- 프로그램에 문제가 발생했을 때 로그를 추가해 원인을 파악하고 해결할 수 있음
- 여기서는 사용자가 로그를 작성하는 방법을 알아보자.
- 질문 목록 조회 시 GET 방식으로 요청되는 page, kw 매개변수의 입력값을 로그로 출력해 보자
- STS로 돌아가 src/main/java 디렉터리의 com.mysite.sbb.question 패키지에 QustionController.java 파일을 다음과 같이 수정해 보자

사용자 로그 작성하기

• QuestionController.java

```
package com.mysite.sbb.question;

(... 생략 ...)
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@RequiredArgsConstructor
@Controller
@RequestMapping("/question")
public class QuestionController {

    (... 생략 ...)
```

```
    @GetMapping("/list")
    public String list(Model model, @RequestParam(value = "page", defaultValue =
"0") int page, @RequestParam(value = "kw", defaultValue = "") String kw) {
        log.info("page:{}, kw:{}, page, kw);
        Page<Question> paging = this.questionService.getList(page, kw);
        model.addAttribute("paging", paging);
        model.addAttribute("kw", kw);
        return "question_list";
    }

    (... 생략 ...)
}
```

· 사용자 로그 작성하기

- 롬복이 제공하는 @Slf4 애너테이션을 사용하면 log 객체를 사용할 수 있음
- log 객체를 사용하여 debug, error 등의 로그 레벨로 로그를 출력할 수 있음
- 로그 레벨(log level)은 다음과 같이 6단계로 구성되며,
각 단계의 로그는 log.trace, log.debug, log.info, log.warn, log.error, log.fatal과
같이 출력할 수 있음

사용자 로그 작성하기

- trace(1단계): 가장 낮은 로그 레벨이며, debug보다 정보를 훨씬 상세하게 기록할 경우에 사용한다.
- debug(2단계): 디버깅 목적으로 사용한다.
- info(3단계): 주요 이벤트나 상태 등의 일반 정보를 출력할 목적으로 사용한다.
- warn(4단계): 문제가 발생할 가능성이 있는 상태나 상황 등(비교적 작은 문제)에 관한 경고 정보를 출력할 목적으로 사용한다.
- error(5단계): 심각한 문제나 예외 상황 등(비교적 큰 문제)에 대한 오류 정보를 출력할 목적으로 사용한다.
- fatal(6단계): 가장 높은 로그 레벨이며, 프로그램 기능의 일부가 실패하거나 오류가 발생하는 등 아주 심각한 문제에 관한 정보를 출력할 목적으로 사용한다.

설명에서 짐작할 수 있듯이 로그 레벨의 순서는 다음과 같음

TRACE < DEBUG < INFO < WARN < ERROR < FATAL

· 사용자 로그 작성하기

- 만약 application-prod.properties 파일에 logging.level.root=info로 설정하면 TRACE, DEBUG 로그는 출력되지 않고 INFO 이상의 로그만 출력됨
- 즉, log.trace 또는 log.debug로 출력하는 로그는 출력되지 않고 logging.info, logging.warn, logging.error, logging.fatal로 출력한 로그만 출력된다는 뜻임
- 이와 같이 QuestionController에 로그를 추가하고 새로운 배포 파일을 작성해 서버에 적용하면 추가한 로그가 sbb.log에 출력되는 것을 확인할 수 있음

- 지금까지 SBB 서비스에 접속하려면 브라우저 주소 창에 다음과 같이 고정 IP를 입력해 왔음

`http://43.202.195.94`

- 웹 사이트 대부분은 이런 식으로 IP를 입력해 접속하지 않음

- 고정 IP는 외국의 인터넷 서비스 업체로부터 제공하는 고정 IP를 사용함

`http://pybo.kr`

- 고정 IP 대신 도메인을 사용하는 방법을 알아보자

도메인 구입하기

도메인을 사용하기에 앞서 가장 먼저 사용할 수 있는 도메인인지 확인해야 함.
 다음 URL(<https://whois.kr>)에서 사용할 수 있는 도메인을 검색해 보자.
 예를 들어 'pahkey.co.kr'을 검색하면 다음과 같이 등록되지 않은 도메인이라 나옴



도메인 구입하기

구매했던 도메인을 찾았다면 이를 구매해 보자.
AWS에서도 도메인을 판매함. AWS에서 도메인을 구매한다면
AWS Route 53(<https://console.aws.amazon.com/route53/home>)에 접속하면 됨.

하지만 AWS Route 53은 pybo.kr과 같이 .kr 또는 .co.kr과 같은 도메인은 취급하지
않음.

만약 .kr과 같은 국내 도메인을 구매하고 싶다면 다른 도메인 업체를 알아봐야 하는
데,
구매 방법은 도메인 업체마다 다르므로 여기서는 다루지 않음

도메인 구입하기

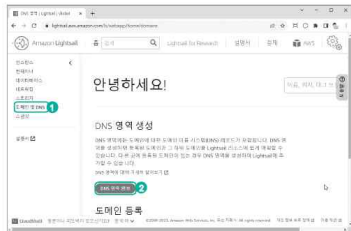
도메인 구입 시에 네임 서버 주소를 설정하는 항목이 있는데 일단 해당 업체의 네임 서버 주소를 사용하는 것으로 설정하자.

네임 서버 주소는 고정 IP와 도메인을 연결하는 역할을 하므로 중요함.
도메인에 등록된 네임 서버 주소는 언제든지 수정할 수 있으니 걱정하지 말자

도메인 연결하기

도메인을 구입했다면 이제 도메인과 AWS에 등록된 고정 IP를 연결해 보자.
그래야만 고정 IP 대신 도메인 주소를 사용할 수 있음

1. <https://lightsail.aws.amazon.com>에 접속해 로그인한 뒤,
[도메인 및 DNS] 메뉴를 선택하고
[DNS 영역 생성]을 선택함



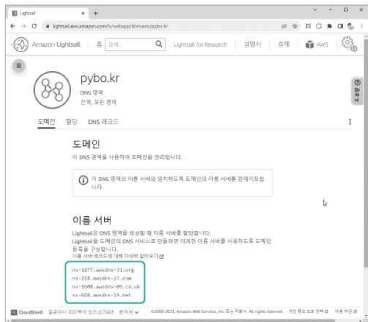
도메인 연결하기

2. 구매한 도메인(예: pybo.kr)을 입력하고
[DNS 영역 생성] 버튼을 클릭함



도메인 연결하기

1. [DNS 영역 생성] 버튼을 클릭하면 다음과 같이 pybo.kr DNS 영역이 생성된다.



도메인 연결하기

[할당] 탭을 선택한 후 [+할당 추가]를 선택함



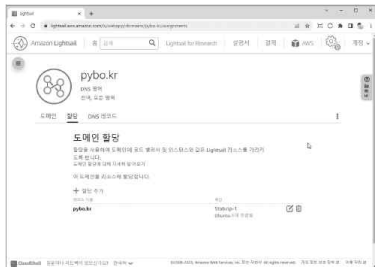
도메인 연결하기

[도메인 이름 선택]에서 pybo.kr와 같이 여러분이 등록한 도메인명을 선택하고,
[리소스 선택]에서 Staticip-1과 같이 이전에 등록한 고정 IP를 선택함.
마지막으로 오른쪽 하단의 [할당]을 클릭함



도메인 연결하기

그러면 다음 화면을 볼 수 있음



도메인 연결하기

이제 AWS 라이트세일에서 네임 서버 주소를 생성했으므로 다시 도메인을 구매한 업체 URL에 접속하여 구매한 도메인과 연결된 네임 서버 주소를 변경해야 함.

다음은 pybo.kr의 네임 서버 주소를 등록하는 화면임. 업체마다 차이는 있겠지만 다음과 같이 AWS Lightsail의 이름 서버(네임 서버) 주소 4개를 차례로 등록하면 됨

이와 같은 곳에 차례로 네임 서버를 등록한다.

도메인 적용하기

도메인이 생성되었으므로 웹 서버의 설정을 변경해야 함.
mobaXterm으로 돌아가 나노 편집기를 통해 엔진엑스의 설정을 변경해 보자

```
• /etc/nginx/sites-available/sbb

server {
    listen 80;
    server_name pybo.kr;

    location / {
        proxy_pass http://localhost:8080;
        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;
    }
}
```

server_name을 localhost에서 pybo.kr로 변경함

도메인 적용하기

2. `server_name`을 수정했으므로 나노 편집기에서 수정 내용을 저장한 후 빠져나와, 엔진엑스를 다시 시작해 보자

```
ubuntu@jumpo:/etc/nginx/sites-enabled$ sudo systemctl restart nginx
```

1. 이제 고정 IP 대신 여러분의 도메인으로 SBB 서비스에 접속할 수 있음



- 이제 브라우저에서 고정 IP 대신 도메인을 입력해 SBB 서비스에 접속할 수 있음
- 하지만 브라우저의 주소 창을 보면 다음과 같이 '주의 요함'이라는 경고 메시지가 표시 됨



- 이러한 경고 메시지가 보이는 이유는 https://이 아닌 주소를 사용하기 때문임
- 즉, https가 아닌 http 프로토콜을 사용했기 때문임
- 브라우저는 HTTPS가 아닌 HTTP 사용 시 항상 이러한 경고 메시지를 보여 줌

HTTPS가 필요한 이유

- HTTPS가 아닌 HTTP 프로토콜을 사용하면
웹 서버와 웹 브라우저 사이에 주고받는 데이터가 암호화되지 않으므로
이 과정에서 누군가가 데이터를 훔쳐볼 수도 있음
- 이 말은 SBB 서비스의 보안이나 개인 정보가 보호되지 않는다는 의미임
- 따라서 웹 브라우저와 SBB 서비스 사이의 네트워크 구간에서
주고받는 데이터는 반드시 암호화하여
데이터가 노출되더라도 무슨 내용인지 알 수 없도록 해야 함

· HTTPS가 필요한 이유

- 이러한 역할을 하는 것이 바로 HTTP에 SSL(Secured Socket Layer) 기능을 더한 HTTPS 프로토콜임
- SBB 서비스에 HTTPS 프로토콜을 제공하려면 SSL 인증서가 필요함
- SSL 인증서를 발급받아 엔진엑스에 적용하여 HTTPS 프로토콜로 서비스를 제공해 보자

SSL 인증서를 발급받아 HTTPS로 접속하기

- SSL 인증서는 인증 기관으로부터 발급받아야 함
- 대표적인 인증 기관으로는 Comodo, Thawte, GeoTrust, DigiCert 등이 있음
- 사실 도메인 구입과 마찬가지로 이러한 인증 기관에서 발급받는 SSL 인증서는 유료임 (인증서의 가격은 인증 기관별로 다르다).
- 하지만 무료로 SSL 인증서를 발급해 주는 Let's Encrypt 서비스를 사용해 보자

SSL 인증서를 발급받아 HTTPS로 접속하기

터미널 프로그램에서 다음과 같이 certbot과 python3-certbot-nginx를 설치해 보자. 이는 각각 Let's Encrypt SSL 인증서를 쉽게 생성하고 관리할 수 있는 도구와 라이브러리임

```
ubuntu@jumpo:~$ sudo apt install certbot  
ubuntu@jumpo:~$ sudo apt install python3-certbot-nginx
```

certbot은 SSL 인증서를 생성, 발급, 갱신, 관리하는 데 사용함.
python3-certbot-nginx는 certbot을 엔진엑스 웹 서버와 함께 사용하려면 필요함

SSL 인증서를 발급받아 HTTPS로 접속하기

- 설치를 완료했다면 엔진엑스 웹 서버에서 사용할 Let's Encrypt의 인증서를 발급하자

```
ubuntu@jumpo:~$ sudo certbot certonly --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): pahkey@gmail.com
```

```
-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
```

```
(Y)es/(N)o: Y
```

```
-----
Would you be willing, once your first certificate is successfully issue, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that deve
lops Certbot? We'd like to send you email about our work encrypting the web, EFF
news, campaigns, and ways to support digital freedom.
-----
```

```
(Y)es/(N)o: Y
Account registered.
Please enter in your domain name(s) (comma end/or space separated) (Enter 'c' to
cancel): pybo.kr ← 도메인 입력
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for pybo.kr
Using default address 80 for authentication.
Waiting for verification...
```

```
(... 생략 ...)
```


SSL 인증서를 발급받아 HTTPS로 접속하기

이와 같이 순서대로 내용을 입력하면 다음과 같은 위치에 인증서가 생성됨

```
/etc/letsencrypt/live/pybo.kr/fullchain.pem  
/etc/letsencrypt/live/pybo.kr/privkey.pem
```

반드시 여러분의 도메인을 입력해야 한다.

SSL 인증서를 발급받아 HTTPS로 접속하기

3.

이제 생성한 SSL 인증서를 엔진엑스에 적용하기 위해 나노 편집기에서 sbb 파일을 수정해 보자.

```
server {
    listen 80;
    server_name pybo.kr;
    rewrite ^ https://$server_name$request_uri? permanent;
}

server {
    listen 443 ssl;
    server_name pybo.kr;

    ssl_certificate /etc/letsencrypt/live/pybo.kr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/pybo.kr/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;

    location / {
        proxy_pass http://localhost:8080;
        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;
    }
}
```

• /etc/nginx/sites-available/sbb

SSL과 관련된 설정을 적용한다.

HTTP 요청을 받는 80번 포트를 HTTPS 요청을 받는 443번 포트로 리다이렉트하도록 설정함.
그리고 설치한 SSL 인증서를 적용하기 위해 SSL 관련 설정을 적용함

SSL 인증서를 발급받아 HTTPS로 접속하기

이와 같이 엔진엑스의 설정을 바꿨으므로 엔진엑스를 재시작해 보자

```
ubuntu@jump10:~$ sudo systemctl restart nginx.service
```

엔진엑스에 SSL을 적용하면
443번 포트의 방화벽을 해제해야 함

IPv4 방화벽 ⑦

인터넷 또는 특정 IPv4 주소 또는 범위에 대해 포트를 개방하는 규칙을 생성합니다.
방화벽 규칙에 대해 자세히 알아보기

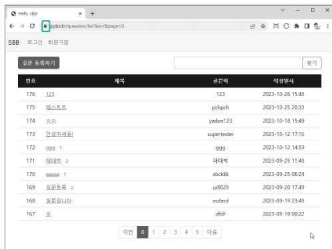
+ 규칙 추가

애플리케이션	프로토콜	포트 또는 범위 / 코드	제한		
SSH	TCP	22	모든 IPv4 주소 LightSail 브라우저 SSH/RDP ⑧	<input checked="" type="checkbox"/>	<input type="checkbox"/>
HTTP	TCP	80	모든 IPv4 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>
HTTPS	TCP	443	모든 IPv4 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>
사용자 지정	TCP	8080	모든 IPv4 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>

SSL 인증서를 발급받아 HTTPS로 접속하기

이제 브라우저에서 `http://pybo.kr` 대신 `https://pybo.kr`로 접속할 수 있음.

그리고 이전에 표시되던
'주의 요함' 경고 메시지도 사라지고
SSL 인증이 되었음을 의미하는
자물쇠 모양의 아이콘도 표시될 것임



번호	아이디	이름	가입일
176	123	123	2023-10-26 15:43
175	test@pybo.kr	pybo.kr	2023-10-25 20:33
174	pybo.kr	pybo.kr	2023-10-18 15:43
173	pybo.kr	pybo.kr	2023-10-12 17:55
172	pybo.kr	pybo.kr	2023-10-12 14:59
171	pybo.kr	pybo.kr	2023-10-25 15:43
170	pybo.kr	pybo.kr	2023-10-25 08:21
169	pybo.kr	pybo.kr	2023-10-20 17:43
168	pybo.kr	pybo.kr	2023-10-19 23:43
167	pybo.kr	pybo.kr	2023-10-19 00:22

SSL 인증서를 발급받아 HTTPS로 접속하기

- H2 데이터베이스는 웹 프로그램이나 서비스 개발 단계에서는 유용하지만 실제 운영 환경에서 사용하기에는 많이 부족한 DBMS임
- 따라서 웹 프로그램이나 서비스를 본격적으로 운영하기로 마음먹었다면 H2보다 성능이 좋은 DBMS를 고려해야 함
- 오라클과 같은 상용 DBMS도 있지만 규모가 작은 웹 서비스는 대체로 PostgreSQL이나 MySQL 등의 무료 DBMS를 주로 사용함
- 이번에는 그중에서도 점유율이 점점 높아지는 PostgreSQL을 사용하는 방법을 알아보자

PostgreSQL 설치하기

- PostgreSQL을 사용하는 방법에는 두 가지가 있음
- 하나는 AWS 서버에 PostgreSQL을 직접 설치해 사용하는 방법이고,
다른 하나는 AWS가 제공하는 데이터베이스 인스턴스를 사용하는 방법임
- 하지만 DB를 공부할 목적이 아니라면
PostgreSQL을 직접 설치하는 것은 권장하지 않음
- PostgreSQL을 설치해 보자

4-12 PostgreSQL로 전환하기

PostgreSQL 설치하기

1. <https://lightsail.aws.amazon.com>에 접속해 로그인한 뒤,
[데이터베이스] 탭을 선택하고
[데이터베이스 생성] 버튼을 클릭함



1. 'AWS 리전 및 가용 영역 변경'을 클릭함



4-12 PostgreSQL로 전환하기

PostgreSQL 설치하기

1. 리전으로 다음과 같이 '서울'을 선택함



1. 스크롤을 내려 다음과 같이 'PostgreSQL'을 선택함



4-12 PostgreSQL로 전환하기

PostgreSQL 설치하기

5. 스크롤을 내려
다음과 같이
\$15 플랜을 선택함



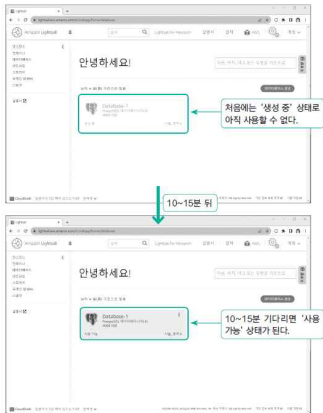
1. 리소스 이름에
'Database-1'을 입력하고
[데이터베이스 생성] 버튼을
클릭함



4-12 PostgreSQL로 전환하기

PostgreSQL 설치하기

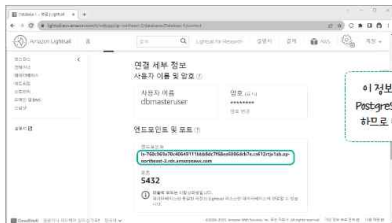
7. 그러면 다음과 같이
데이터베이스가 생성됨



4-12 PostgreSQL로 전환하기

PostgreSQL 설치하기

사용 가능한 상태가 되면 'Database-1'을 선택해 보자.
그러면 다음과 같은 '연결 세부 정보'를 확인할 수 있는 화면이 나타남



PostgreSQL 설치하기

· '연결 세부 정보'에서는 중요한 정보 3가지를 확인할 수 있음

- 사용자 이름: 사용자 이름으로 dbmasteruser를 확인할 수 있다.
- 암호: '표시'를 클릭하면 암호를 볼 수 있다.
- 데이터베이스 주소: '엔드포인트'로 데이터베이스의 도메인 주소가 적혀 있다.

데이터베이스 생성하기

AWS의 PostgreSQL 인스턴스를 생성함.

이제 생성된 PostgreSQL 인스턴스에 SBB 서비스에 사용할 sbb 데이터베이스를 생성해보자

PostgreSQL에 접속하려면 다음과 같이 PostgreSQL 클라이언트를 먼저 설치해야 함.
PostgreSQL 클라이언트는 PostgreSQL 데이터베이스 서버와 상호 작용을 하는 소프트웨어 또는 프로그램이라고 생각하면 됨

```
ubuntu@jumpo:~$ sudo apt install postgresql-client
```

데이터베이스 생성하기

postgresql-client 패키지를 설치하면 PostgreSQL에서 새로운 DB를 생성할 때 사용하는 createdb나 DB를 연결하고 DB와 상호 작용 할 때 사용하는 psql 등의 명령어를 터미널에서 사용할 수 있음

설치를 완료한 후, createdb 명령을 실행하여 sbb 데이터베이스를 생성해 보자

```
ubuntu@jumpo:~ $ createdb sbb --username=dbmasteruser -h ls-  
be78fd2c2exxxxxxxxxxxxxxxxxx2c9.cq1cyugj7ibs.ap-northeast-2.rds.amazonaws.com
```

↑
강조한 부분이 엔드포인트에 해당
하는 DB 주소이다.

createdb 명령을 실행하면 암호를 물어보는데 '연결 세부 정보'의 암호를 입력하면 됨

데이터베이스 변경하기

SBB가 PostgreSQL 서버에 접속하려면 PostgreSQL 라이브러리가 필요함.
다음과 같이 H2 데이터베이스를 설치할 때와 마찬가지로
build.gradle 파일을 수정하여 PostgreSQL 라이브러리를 설치해 보자

```
• build.gradle

(... 생략 ...)

dependencies {
    (... 생략 ...)
    implementation 'org.commonmark:commonmark:0.21.0'
    runtimeOnly 'org.postgresql:postgresql'
}

(... 생략 ...)
```

데이터베이스 변경하기

- 그다음 build.gradle 파일을 선택한 후 마우스 오른쪽 버튼을 눌러 [Gradle → Refresh Gradle Project]를 클릭하여 필요한 라이브러리를 설치하자
- 로컬 서버는 기존의 H2 데이터베이스를 그대로 사용하고
운영 서버만 PostgreSQL 데이터베이스를 사용해야 하므로
src/main/resources 디렉터리의 application-prod.properties 파일을
다음과 같이 수정해 보자

데이터베이스 변경하기

2.

```
• application-prod.properties

# DATABASE
spring.datasource.url=jdbc:postgresql://<데이터 베이스 주소>:5432/sbb
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.username=dbmasteruser
spring.datasource.password=<암호>

# JPA
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show_sql=true

# logging
(... 생략 ...)
```

<데이터베이스 주소>와 <암호>는 터미널 프로그램에서 입력했던 것과 마찬가지로
엔드포인트에 해당했던 DB 주소와 암호를 입력함

데이터베이스 변경하기

과정을 모두 마쳤다면 새로운 배포 파일을 생성하여 서버에 적용하고 stop.sh, start.sh를 실행하여 SBB 서비스를 재시작하자.

데이터베이스가 변경되었으므로 기존 데이터는 보이지 않을 것임

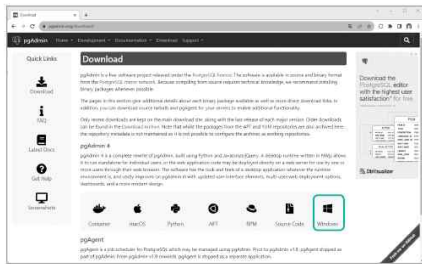
· PostgreSQL 접속하기

- H2 데이터베이스의 GUI 도구로 H2 Console이 있다면 PostgreSQL에는 pgAdmin이 있음
- 로컬 PC에 pgAdmin을 설치하여 PostgreSQL 서버에 접속해 보자

4-12 PostgreSQL로 전환하기

PostgreSQL 접속하기

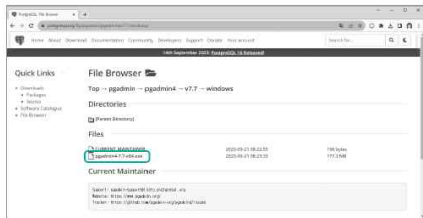
이어서 데이터베이스를 관리하는 GUI 도구인 pgAdmin을
<https://www.pgadmin.org/download>에서 pgAdmin을 내려받아 설치해 보자



4-12 PostgreSQL로 전환하기

PostgreSQL 접속하기

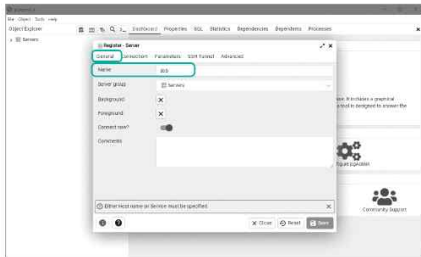
1. 연결된 내려받기 페이지에서 가장 최신 버전인 pgAdmin 4를 선택한 후, 'Files'에서 pgadmin4.x-x64.exe를 클릭해 파일을 내려받아 설치하자. 이후 과정은 다른 프로그램 설치 과정과 동일하므로 생략함



4-12 PostgreSQL로 전환하기

PostgreSQL 접속하기

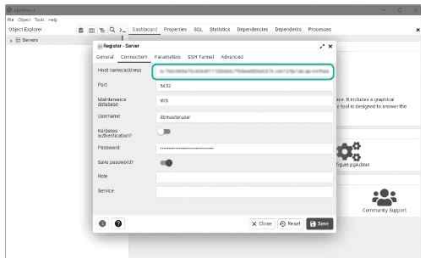
메인 화면에서 'Add New Server'를 클릭하면 다음 화면이 등장함.
[General] 탭의 'Name' 항목에 'sbb'를 입력함



4-12 PostgreSQL로 전환하기

PostgreSQL 접속하기

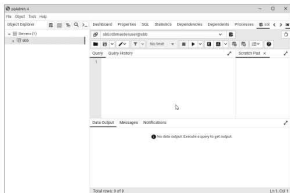
[Connection] 탭을 선택하고 데이터베이스 주소, 포트 번호, 데이터베이스명, 사용자 이름, 암호를 순서대로 입력하고 [Save] 버튼을 눌러 저장해 보자



4-12 PostgreSQL로 전환하기

PostgreSQL 접속하기

다음은 생성한 SBB 데이터베이스에 접속한 모습임.



그동안 H2 데이터베이스와 H2 콘솔에서 했던 작업을 PostgreSQL과 pgAdmin에서 할 수 있음.
Query Editor에서 쿼리문을 작성하면 아래 Data Output을 통해 원하는 데이터나 결과가 출력될 것임