



# 섹션 6. 생애 최초 배포 준비하기

## 이번 섹션의 목표

37강. 배포란 무엇인가

38강. profile과 H2 DB

39강. git과 github이란 무엇인가?!

40강. git 기초 사용법

41강. AWS의 EC2 사용하기

42강. Section 6 정리. 다음으로!

## 이번 섹션의 목표

1. 배포가 무엇인지 이해하고, 배포를 하기 위해 어떤 준비를 해야 하는지 알아본다.
2. 스프링 서버를 실행할 때 DB와 같은 설정들을 코드 변경 없이 제어할 수 있는 방법을 알아본다.
3. git과 github의 차이를 이해하고 git에 대한 기초적인 사용법을 알아본다.
4. AWS의 EC2가 무엇인지 이해하고, AWS를 통해 클라우드 컴퓨터를 빌려 본다.

## 37강. 배포란 무엇인가



강의 영상과 PPT를 함께 보시면 더욱 좋습니다 😊

이번 시간에는 ‘배포’가 무엇인지 알아보자. 위키 백과에서 배포의 정의를 찾아보면, 최종 사용자에게 SW를 전달하는 과정이라고 나온다. 어렵뜻이 알 것 같긴 한데, 정확히는 모르겠다. 우선 우리가 만든 도서관리 애플리케이션의 현재 상황을 살펴보자.



현재는 Spring과 MySQL을 이용해 개발한 도서관리 애플리케이션 서버가 우리 컴퓨터에서 실행되고 종료되고 있다. 이제 이 서버를 우리 친구가 사용해야 한다고 하자. 우리의 친구는 어떻게 도서관리 애플리케이션을 사용할 수 있을까?

가장 먼저 생각나는 첫 번째 방법은 친구가 직접 집으로 놀러 오는 것이다. 하지만~ 조금만 생각해 보면 당연히 이상하고 불편하다. 항상 사용할 수 없을뿐더러 몸을 직접 움직여야 한다.

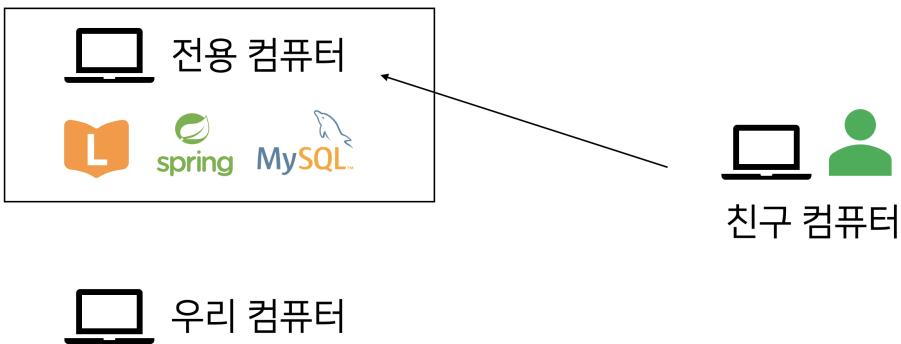
그렇다면 두 번째 방법은, 친구의 컴퓨터를 이용해 우리 컴퓨터에 접속하는 방법이다.



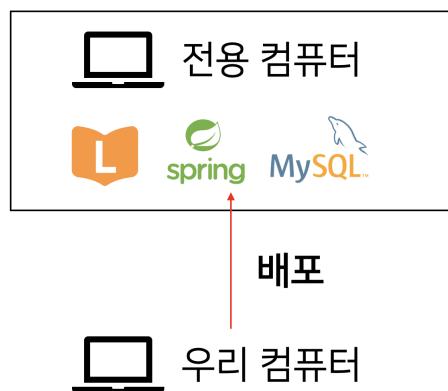
첫 번째 방법보다는 낫지만, 여전히 문제가 존재한다.

1. 우리 컴퓨터는 24시간 켜져 있지 않다. 친구가 원할 때 접속을 못할 수도 있다.
2. 우리 컴퓨터는 스프링, MySQL 외에도 문서 편집기, 게임, 비디오 재생 등을 처리한다. 다른 곳에서 컴퓨터 자원을 많이 사용하고 있는 셈이다.

마지막 세 번째 방법은, 우리가 우리 컴퓨터 대신, 전용 컴퓨터를 가져와 코드를 읽기고, 스프링, MySQL 등을 설치해 친구가 접속하게 하는 방법이다. 이렇게 되면, 친구는 24시간 접속할 수 있고, 전용 컴퓨터는 정말 필요한 프로그램만 실행할 것이다.

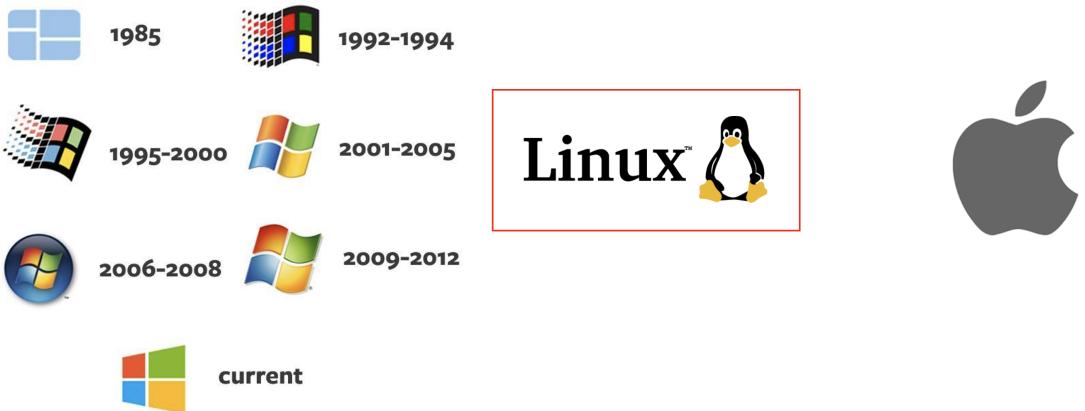


세 번째 방법을 사용하려면, 필요한 프로그램이 설치된 전용 컴퓨터에 우리의 코드를 옮기고 실행시켜야 하는데 이 과정을 바로 배포라 부르게 된다.



자 그런데 한 가지 함정이 있다! 바로 우리는 전용 컴퓨터가 없다는 것이다. 때문에 우리는 미국 쇼핑몰 아마존이 운영하는 AWS(Amazon Web Service)에서 컴퓨터를 빌려야 한다. 최대 1년까지는 작고 귀여운(?) 무료 컴퓨터를 빌릴 수 있으니, 이를 활용하도록 하자.

추가로, AWS에서 컴퓨터를 빌릴 때 한 가지 알아두어야 할 점이 있다. 바로 우리가 컴퓨터를 살 때는 운영체제(OS)도 함께 선택해야 한다는 점이다. 운영체제는 우리가 흔히 사용하는 Windows를 생각하면 된다! 이 Windows 외에도 Linux, Mac OS 등이 대표적인 운영체제로 있는데, 서버용 컴퓨터는 보통 Linux 운영체제를 가진 컴퓨터를 사용하게 된다.

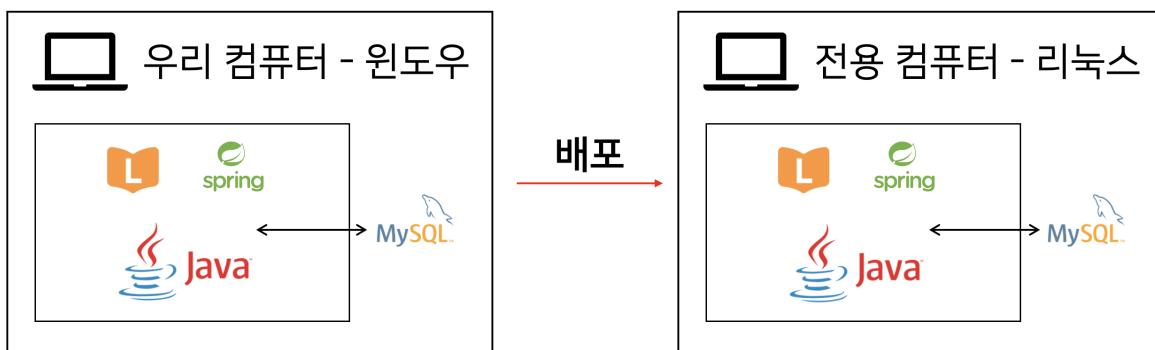


매우 좋다~! 😊 이번 Section에서는 배포를 위한 여러 준비들을 하고, 다음 Section에는 AWS를 이용해 빌린 컴퓨터에 접속해 실제 배포를 해보도록 하자.

## 38강. profile과 H2 DB

이번 시간에는 배포를 하기 위해 필요한 profile이란 개념에 대해 다루고, H2 DB가 무엇이고 왜 사용하는지 알아보도록 하자.

이전 시간에, 배포란 우리 컴퓨터에 있는 코드들을 전용 컴퓨터로 옮기고 java나 MySQL 등을 설치하고 실행하는 것을 의미한다고 다루었다. 이 과정을 조금 더 자세히 살펴보면 다음과 같다.



우리 컴퓨터에서 <도서 관리 애플리케이션>을 실행할 때에는 우리 컴퓨터에 설치했던 java와 mysql을 실행했다. 또한, <도서 관리 애플리케이션> 서버는 우리 컴퓨터에 있는 mysql을 사용했다.

반면, 전용 컴퓨터에 똑같은 환경을 갖추고 실행을 하려면, 리눅스용 java와 리눅스용 mysql을 설치, 실행해야 하며 전용 컴퓨터에서 실행된 서버는 전용 컴퓨터에 있는 mysql을 사용해

야 한다!

핵심은 이렇다. 똑같은 서버 코드를 실행시키지만, 우리 컴퓨터에서 실행할 때에는 우리 컴퓨터의 MySQL을 사용해야 하고, 전용 컴퓨터에서 실행할 때는 전용 컴퓨터의 MySQL을 사용해야 한다.

바꿔 말하자면 똑같은 서버 코드를 실행시키지만, 실행될 때의 설정을 다르게 하고 싶다는 의미이다. 지금은 DB 설정만 존재하지만 서비스가 커지게 되면, 다양한 종류의 설정이 있을 수 있다.

이럴 때 필요한 것이 바로 profile이다! 사실 profile을 지금도 사용하고 있었다. 서버를 실행 시켜 보면 나오는 로그 중 이런 문구가 있다.

```
No active profile set, falling back to 1 default profile: "default"
```

해석해 보면, 활성화된 profile은 없기 때문에 `default` profile을 적용했다는 의미이다. 기본 profile을 사용하고 있던 셈이다.

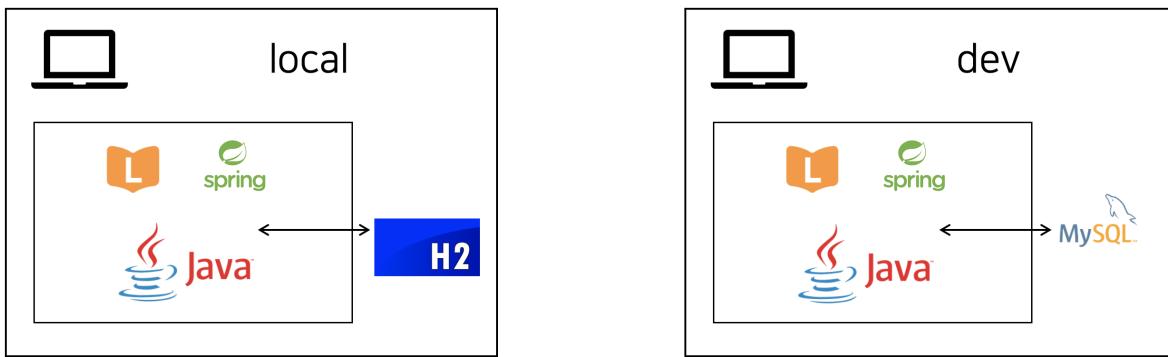
진작 profile을 사용하고 있었다니, 그럼에도 불구하고 아직 profile이 무엇인지 감이 명확히 잡히지는 않는다. 이럴 때는 바로 한번 profile이란 개념을 적용해 보아야 한다!

상황에 따라서 H2라는 DB를 사용하게 하거나, MySQL DB를 사용하게 하자!

H2 DB는 경량 Database로 개발 단계에서 많이 사용하고 있으며, Disk가 아닌 Memory에 데이터를 저장할 수 있다. 이전 10강에서 다루었던 것처럼 Memory에 데이터를 저장하게 되면, 데이터가 휘발되는데 이 때문에 H2 DB는 개발 단계에서만 사용한다. 또한, 개발 단계에서는 테이블에 계속 변경되는데 어차피 데이터가 휘발되기 때문에 `ddl-auto` 옵션을 `create`로 주면 테이블을 신경 쓰지 않고 코드에만 집중할 수 있게 된다!

데이터가 휘발된다는 단점을 코드에만 집중하는 데 사용하는 것이다.

그래서 우리는 local이라는 profile을 입력하게 되면, H2 DB를 사용하고 dev라는 profile을 입력하면 MySQL DB를 사용하게 바꿀 것이다.



`application.yml` 을 다음과 같이 변경해주자!

```

spring:
  config:
    activate:
      on-profile: local
  datasource:
    url: "jdbc:h2:mem:library;MODE=MYSQL;NON_KEYWORDS=USER"
    username: "sa"
    password: ""
    driver-class-name: org.h2.Driver
  jpa:
    hibernate:
      ddl-auto: create
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        dialect: org.hibernate.dialect.H2Dialect
    open-in-view: false
  h2:
    console:
      enabled: true
      path: /h2-console
  ---
spring:
  config:
    activate:
      on-profile: dev
  datasource:
    url: "jdbc:mysql://localhost/library"
    username: "root"
    password: ""
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        show_sql: true

```

```
    dialect: org.hibernate.dialect.MySQL8Dialect
    open-in-view: false
```

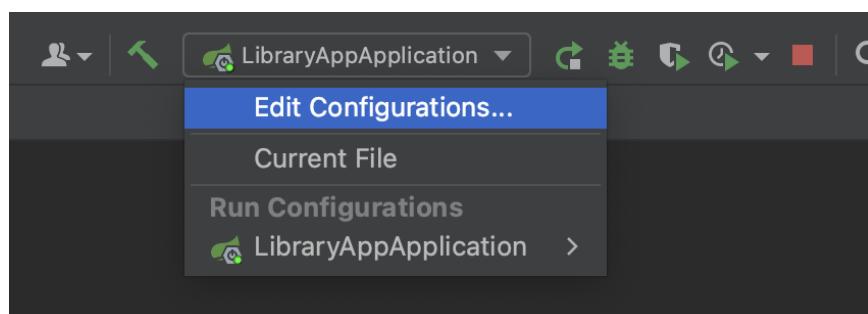
- `spring.config.active.on-profile`
  - 어떤 profile에서 아래 활성화되는 설정인지 지정한다.
  - 예를 들어 `local` profile은 구분선 `---` 위까지의 설정을 가지고 있다.
- `jdbc:h2:mem:library;NON_KEYWORDS=USER`
  - H2를 사용하기 위한 url을 설정해 주었다.
  - 특이사항으로는, USER라는 키워드를 테이블에서 사용할 수 있도록 처리해주었다. 이 설정이 없다면, H2 DB에서는 user라는 키워드가 예약어로 잡혀 있어 사용할 수 없다.
- `datasource.username`
  - H2의 유저 이름을 선택해 주었다. H2는 직접 설치한 게 아니기 때문에 아무 문구나 적어주어도 괜찮다.
  - 잠시 후 H2 web console에 들어갈 때 입력하게 된다.
- `datasource.password`
  - H2의 유저 비밀번호이다. 마찬가지로 잠시 후 H2 web console에 들어갈 때 입력하게 된다.
- `driver-class-name: org.h2.Driver`
  - DB와의 접속을 담당하는 Driver 역시 H2의 Driver를 사용하도록 변경해 주었다.
- `jpa.hibernate.ddl-auto: create`
  - H2 DB를 In-Memory로 설정했기 때문에 서버가 시작될 때마다 테이블을 제거하고 만들어주는 것이 좋다. 그래야 개발 과정에서 테이블이 계속 바뀌더라도 DB를 신경 쓰지 않고 개발할 수 있다.
- `jpa.properties.dialect: org.hibernate.dialect.H2Dialect`
  - H2를 사용하게 바꿔었기 때문에 JPA의 방언 역시 `H2Dialect`로 변경해 주었다.
- `h2.console.enabled: true`
  - H2 web console을 활성화한다. 이제 곧 들어갈 것이다!
- `h2.console.path: /h2-console`
  - 활성화된 H2 web console의 주소를 의미한다. 이 주소로 들어가면 된다!

- 

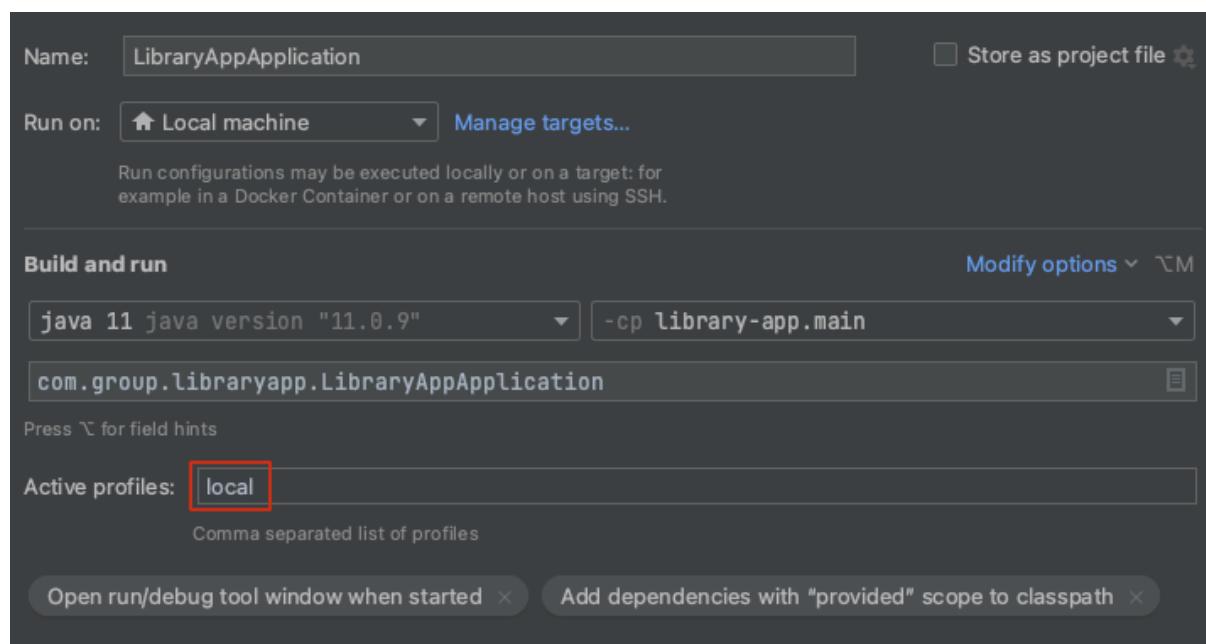
- 구분선이다. 이 구분 선을 기준으로 다른 profile이 적용된다.

매우 좋다~! 😊 우리는 이제 local profile에서는 H2 DB를 사용하도록 하고, dev profile에서는 MySQL DB를 사용하도록 설정했다!

이제 서버를 실행할 때 profile을 변경하는 방법을 알아보도록 하자! IntelliJ 화면의 오른쪽 상단을 보면



`Edit Configurations...` 를 누르면 아래와 같은 창이 나오는데, 이 창에서 `Active profiles` 에 `local` 을 표시해 주면 서버가 local profile을 가진 채로 실행되게 된다!



이제 **OK**를 누르고 서버를 실행 혹은 재실행 해보자! 그러면 아무것도 설정하지 않았을 때와는 다른 로그가 나오게 된다!

```
The following 1 profile is active: "local"
```

local profile이 적용된 채로 서버가 실행된 것이다! 이렇게 되면 H2 DB를 사용한 것이니 설정해 둔 H2 Web console로도 들어갈 수 있다.

<http://localhost:8080/h2-console>에 접속해 보자!

English ▾ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:library;NON\_KEYWORDS=USER

User Name: sa

Password:

Connect Test Connection

접속하면 다음과 같은 화면이 나온다!! 이제 여기에 다음과 같이 설정해두고, **Connect**를 누르자!

- JDBC URL - `jdbc:h2:mem:library`
- User Name - `application.yml`에서 설정했던 유저 이름
- Password - `application.yml`에서 설정했던 비밀번호

이제 **Connect**를 누르면 다음과 같은 창이 나오고, SQL을 마음껏 실행할 수 있다!

The screenshot shows the H2 Database Browser interface. On the left, there's a tree view of database schemas:

- jdbc:h2:mem:library
- + ADDRESS
- + BOOK
- + PERSON
- + USER
- + USER\_LOAN\_HISTORY
- + INFORMATION\_SCHEMA
- + Users

At the bottom of the tree view, it says "H2 2.1.214 (2022-06-13)".

On the right, there's a toolbar with buttons: Run, Run Selected, Auto complete, Clear, and SQL statement:. Below the toolbar, the SQL statement "select \* from user;" is entered.

이어 도서관리 애플리케이션 웹 UI를 열어 데이터를 저장해 보고 정말 H2에 잘 쌓였는지 SQL을 날려 확인해 보자!

이다음에는, profile을 다시 dev로 변경해 서버를 실행해 보자! 그렇게 되면, 다시 MySQL에 있던 데이터가 화면에 보이게 될 것이다!

정말 좋다~!! 이런 방식으로 우리는 우리 컴퓨터에서는 H2를 사용하고, 전용 컴퓨터에서는 전용 컴퓨터의 MySQL을 사용하도록 설정할 수 있다. 물론 우리 컴퓨터에서도 우리 컴퓨터의 MySQL을 사용하도록 할 수 있다. 각자 개발할 때 편한 방식대로 환경을 설정해 보자.

다음 시간에는 git이 무엇이고 왜 사용하는지, git의 기초 사용법은 무엇인지 알아볼 것이다.

## 39강. git과 github이란 무엇인가?!



강의 영상과 PPT를 함께 보시면 더욱 좋습니다 😊

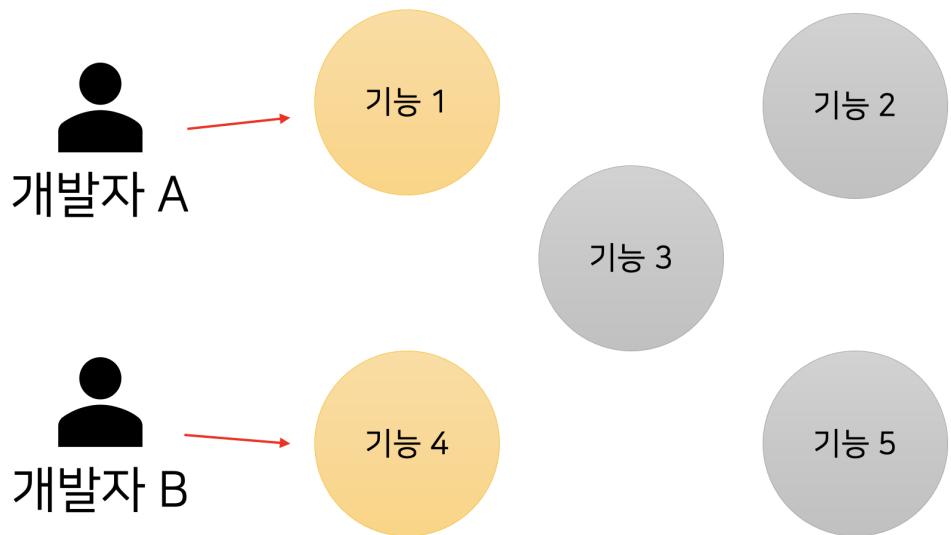
이번 시간에는 git과 github이 무엇인지 알아보고, git의 기초 사용법을 알아보자.

많은 분들께서 개발 공부를 처음 시작해 이런저런 검색을 하다 보면, 아래와 같은 로고를 가진 사이트를 만나볼 수 있다. 바로 github 사이트이다!



때문에 git과 github을 같은 것으로 생각하기도 하는데, 둘은 완전히 다르다! 먼저 git이란 코드를 쉽게 관리할 수 있도록 해주는 버전 관리 프로그램이다. 아하~ 어떤 느낌인지 와닿지 않는다. 다음과 같은 상황을 생각해보자.

우리 회사에는 총 2명의 개발자가 있다. 그리고 정해진 일정 내에 5개의 기능을 개발해야 한다! A 개발자는 기능1을 담당하고 B 개발자는 기능4를 담당하기로 했다! 일정이 빠듯해서 각자 개발을 해야 하는데...

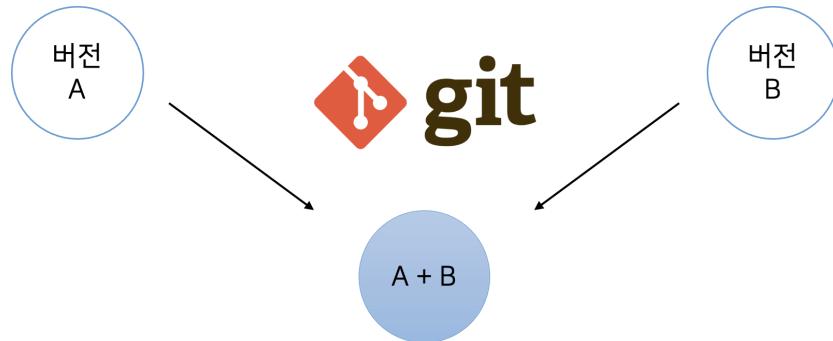


좋다~ 각자 개발하는 것 까지는 문제가 없었다! 그런데 이제 각자 개발한 내용을 서로 합치려 하니 문제가 발생한다! 작업 내용이 많아 수십 개의 파일, 수백 개의 클래스에 걸쳐 있다면, 서로 작업한 것만 복붙할 수도 없고... 만약 같은 코드를 변경했다면 더 어려워진다.



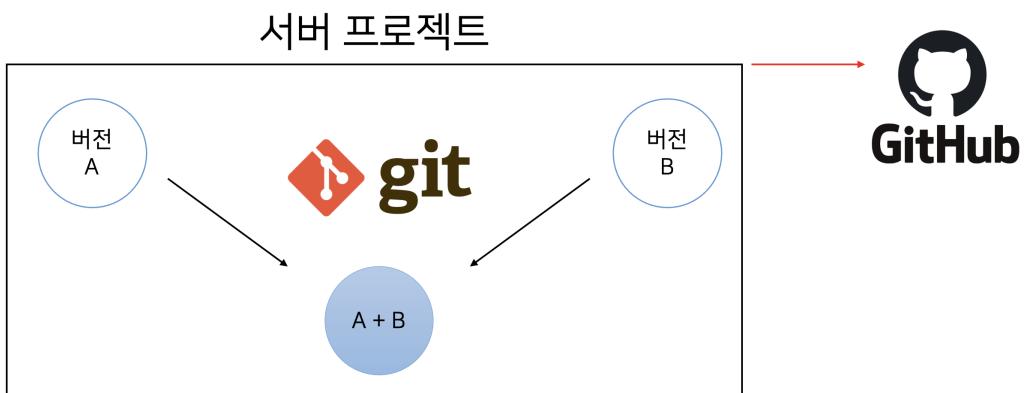
바로 그래서 `git` 이 등장하게 되었다!! 🎉

`git` 은 위와 같은 상황에서 코드를 쉽게 관리할 수 있도록 해주는 버전 관리 프로그램이다. 손쉽게 A 버전, B 버전을 만들 수 있게 해주고, 버전 A와 버전 B를 합치는 데 도움을 주기도 한다.

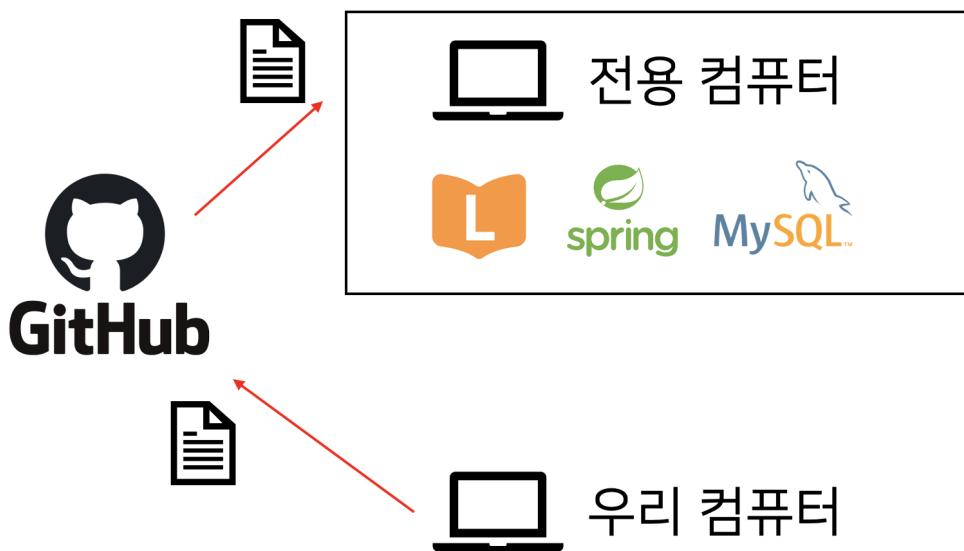


`git` 을 이용하면, 버전 관리를 하기 위해서 기획서\_최종 / 기획서\_최초종 / 기획서\_찐찐최종 처럼 불편한 방법을 쓰지 않아도 되는 것이다!

그렇다면, `github` 이란 무엇인가? `github` 은 `git` 으로 관리되는 프로젝트의 코드가 저장되는 저장소이다! 쉽게 말해 우리가 파일을 구글 드라이브에 보관하듯이, 코드를 `github` 에 보관한다고 생각하면 된다.



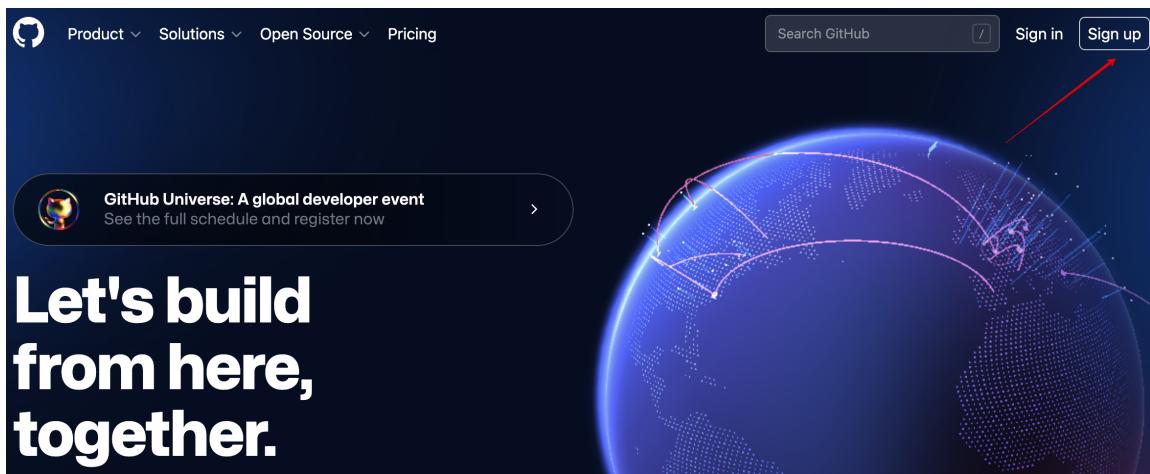
`github`을 이용하게 되면 컴퓨터에 있는 코드가 모종의 이유로 소실되더라도 원본 코드를 지킬 수 있고, 배포를 할 때에도 활용할 수 있다.



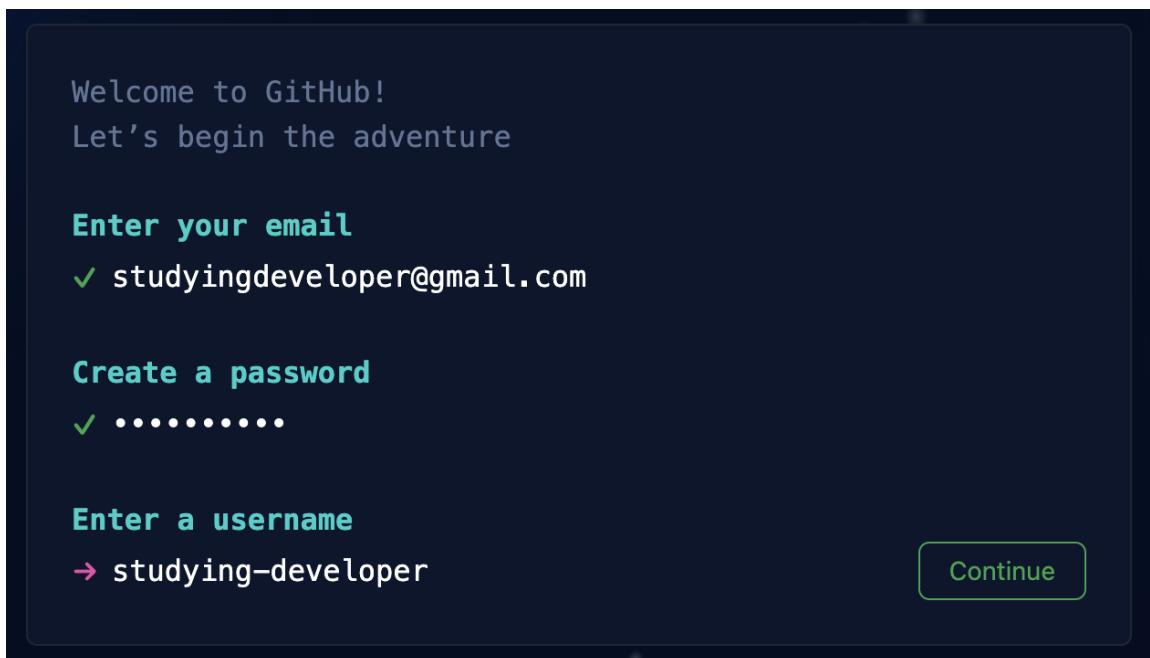
자 이제 다음 시간에는 git을 이용해 github에 우리가 개발한 <도서 관리 애플리케이션>을 업로드해볼 것이다~! 😊 그러기 위해서는 github 계정이 필요하다. github 회원가입 방법은 다음과 같다. 화면의 디자인은 시간이 지나면서 바뀔 수 있지만, 절차는 비슷할 것이다!

#### [github 회원가입 방법]

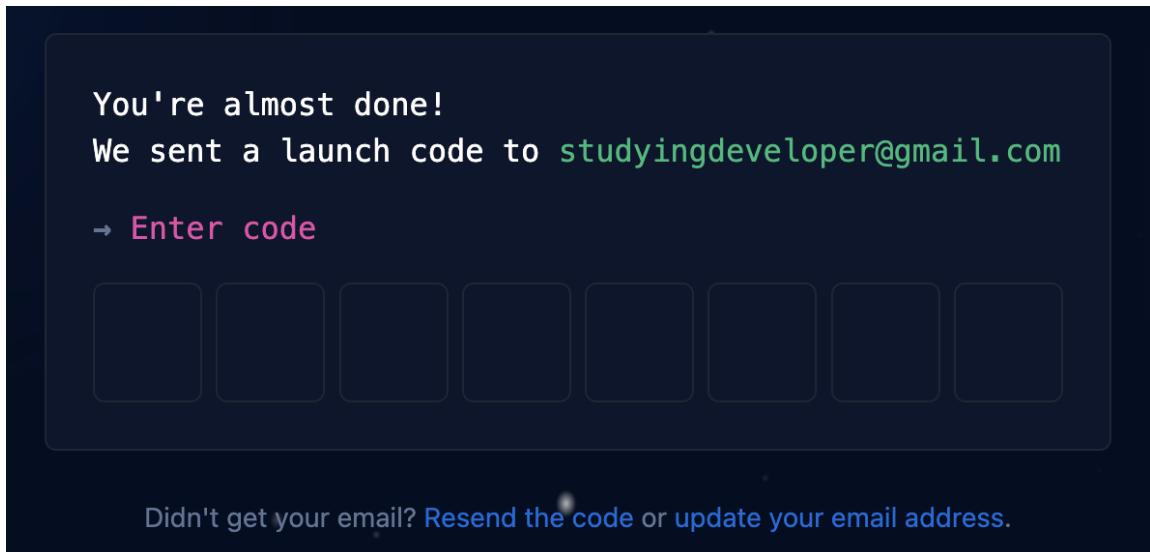
1. <https://github.com/> 사이트에 접속한다.
2. 우측 상단에 회원가입 - `Sign up` 을 누른다



3. 이메일과 비밀번호, 별명을 입력한다.



4. 로봇이 아닙니다 인증을 하고 계정을 생성한다! 그러면 다음과 같이 메일을 인증하라고 한다.

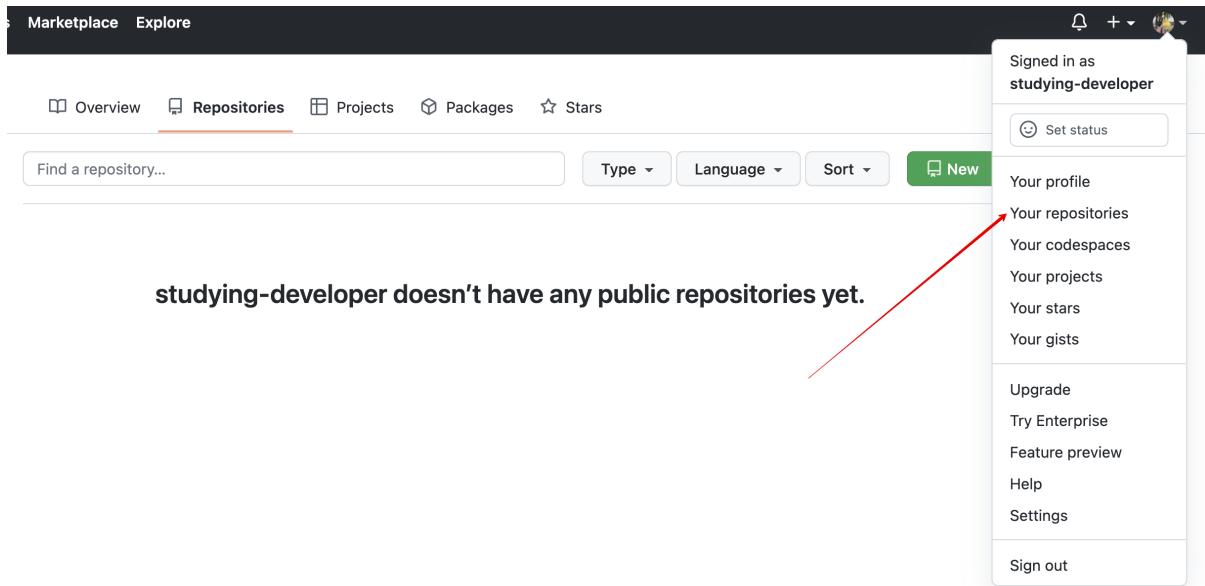


5. 메일에 온 숫자 코드를 입력하면, 회원가입이 완료되고 간단한 설문을 하게 된다. [skip personalization](#) 을 누르면 설문은 패스할 수 있다.
6. 이제 회원가입이 완료되었다!!!

## 40강. git 기초 사용법

이번 시간에는 git을 이용해 github에 우리의 프로젝트를 업로드할 것이다!

먼저 지난 시간에 가입했던 github에 접속해, 코드 저장소 목록에 들어가자. 저장소 목록은 github 사이트를 접속해 오른쪽 상단에 있는 버튼을 눌러 Your repositories를 고르면 목록을 확인할 수 있다!



지금은 당연히 저장소 목록을 만들지 않았기 때문에 비어 있다. 이제 화면에 보이는 `New` 버튼을 눌러주자.

그러면 다음과 같이 저장소를 만드는 화면이 등장한다. 우리는 저장소의 이름과 설명을 간단하게 적어주고, `Create repository` 를 눌러주자!

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner \* Repository name \*

 studying-developer | library-app ✓

Great repository names are short and memorable. Need inspiration? How about [effective-disco](#)?

Description (optional)  
간단 도서관리 애플리케이션

---

 Public  
Anyone on the internet can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

Add a README file  
This is where you can write a long description for your project. [Learn more](#).

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

---

ⓘ You are creating a public repository in your personal account.

**Create repository**

그러면, 다음과 같은 여려 영어가 나오게 된다..!! 아직은 괜찮다~~ 자 이제 원격 저장 공간은 만들어 두었다. 우리가 가지고 있는 코드를 업로드하면 된다. IntelliJ로 돌아가자.

**Quick setup — if you've done this kind of thing before**

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/studying-developer/library-app.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

**...or create a new repository on the command line**

```
echo "# library-app" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/studying-developer/library-app.git
git push -u origin main
```

**...or push an existing repository from the command line**

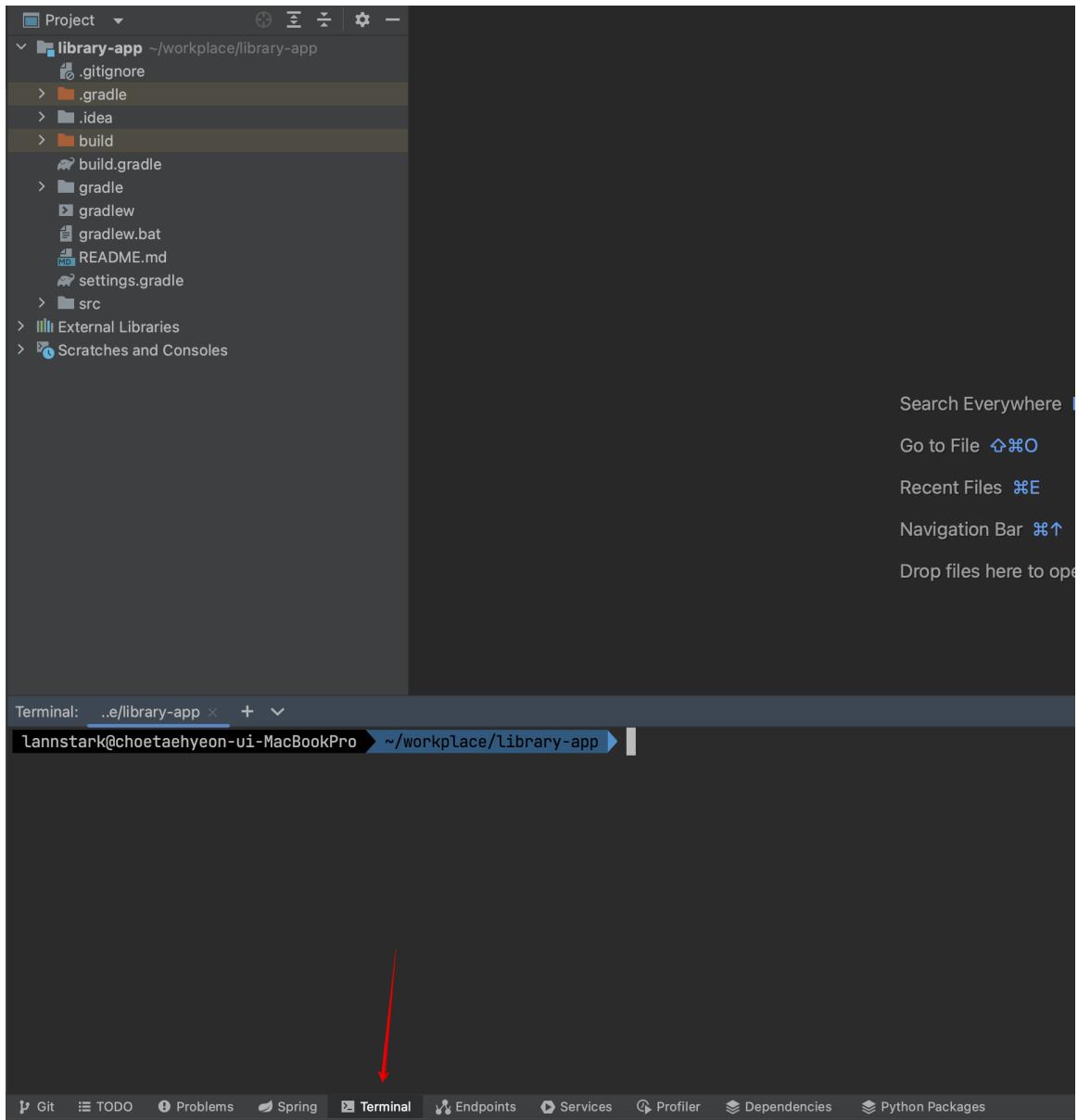
```
git remote add origin https://github.com/studying-developer/library-app.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

설치 영상에서 git을 미리 준비해두었기 때문에, IntelliJ IDEA에 있는 Terminal을 이용해서 git 명령어를 타이핑할 수 있다! 👍



이제 우리는 git을 이용해 코드의 버전을 관리하고 원격 저장소에 코드를 업로드할 것이다.

가장 최초로 타이핑해 주어야 할 명령어는 `git init` 이다.

`git init`은 “이 프로젝트를 이제 git이 관리하겠다”라는 의미이다.

다음으로는 `git remote add origin [각자의 주소]` 이다. 각자의 주소에는 아까 저장소를 만들었을 때 나온 `https` 주소를 넣어주면 된다!

예를 들어, `git remote add origin https://github.com/studying-developer/library-app.git` 를 타이핑해 주면 된다. 이 명령어는 우리의 도서관리 애플리케이션 프로젝트의 git 저장소를 주어진 주소로 하겠다~ 라는 의미이다!

매우 좋다~! 😊 이제 기초적인 세팅은 모두 끝이 났고, git 명령어를 활용해 지금 코드를 github 저장소에 저장할 것이다. 이 과정은 택배를 보내는 과정과 유사하다! 우리가 택배를 보내는 것을 생각해 보면 다음과 같은 과정을 거친다.

1. 택배 상자를 가져와 물건을 담는다.
2. 택배 상자를 포장하고 송장을 붙인다. 이때 하고 싶은 말도 적을 수 있다.
3. 택배 상자를 우체국에 가서 부친다.

비슷하다~ 하나씩 해보도록 하자.

우선 우리는 이 코드들을 택배 상자에 담아야 한다. 이때 명령어는 `git add .`이다. `.`의 의미는 모든 파일을 뜻한. 만약 특정 파일만 택배 상자에 담고 싶다면, `git add 파일 이름`을 타이핑하면 된다.

택배 상자에 잘 담겼는지 확인하는 명령어는 `git status`이다. 타이핑 해보면, 초록색으로 파일들이 나오며, 실제 잘 담겨 있음을 알 수 있다. 만약 특정 파일을 조금 변경하고 다시 한번 `git status`를 타이핑하면 그 변경은 택배 상자에 담기지 않았기 때문에 빨간색으로 나오는 것을 확인할 수 있다.

이제 파일들을 포장하고 송장을 붙여야 한다. 그 명령어는 `git commit -m "메시지"`이다. 메시지에는 함께 적고 싶은 문장을 적으면 된다. 예를 들면 `git commit -m "첫 번째 커밋"`이라고 타이핑할 수 있다.

이때 다음과 같은 문구가 나올 수 있다!

```
*** Please tell me who you are.  
  
Run  
  git config --global user.email "you@example.com"  
  git config --global user.name "Your Name"
```

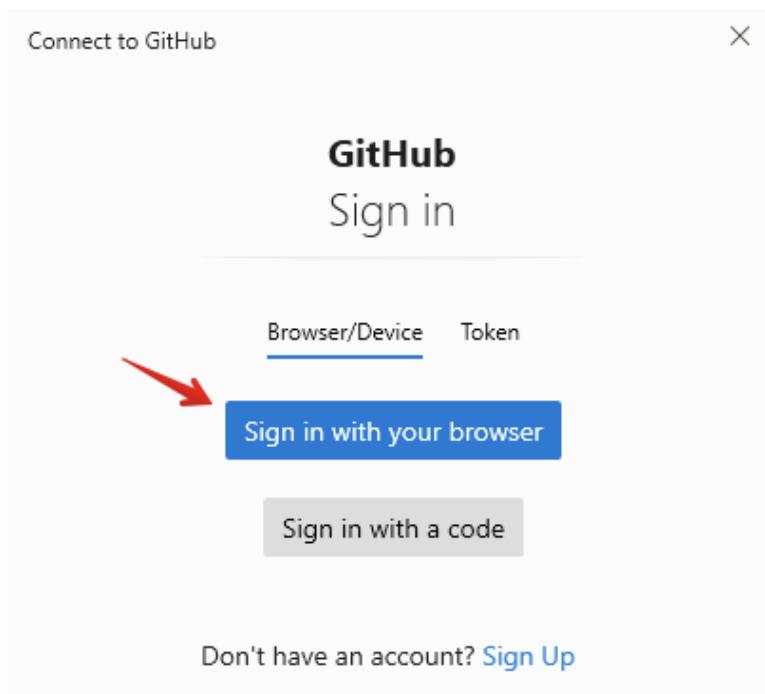
이 문구는 컴퓨터에서 처음 git commit을 하게 되면 나오는데, 커밋을 할 때, 즉 송장을 붙일 때 누가 송장을 붙였는지 추가 기록을 하기 위해서이다. 이 문구를 봤다면, github에 회원가입할 때 사용했던 이메일과 이름을 비슷하게 잘 입력해 주도록 하자! 예를 들면 다음과 같다.

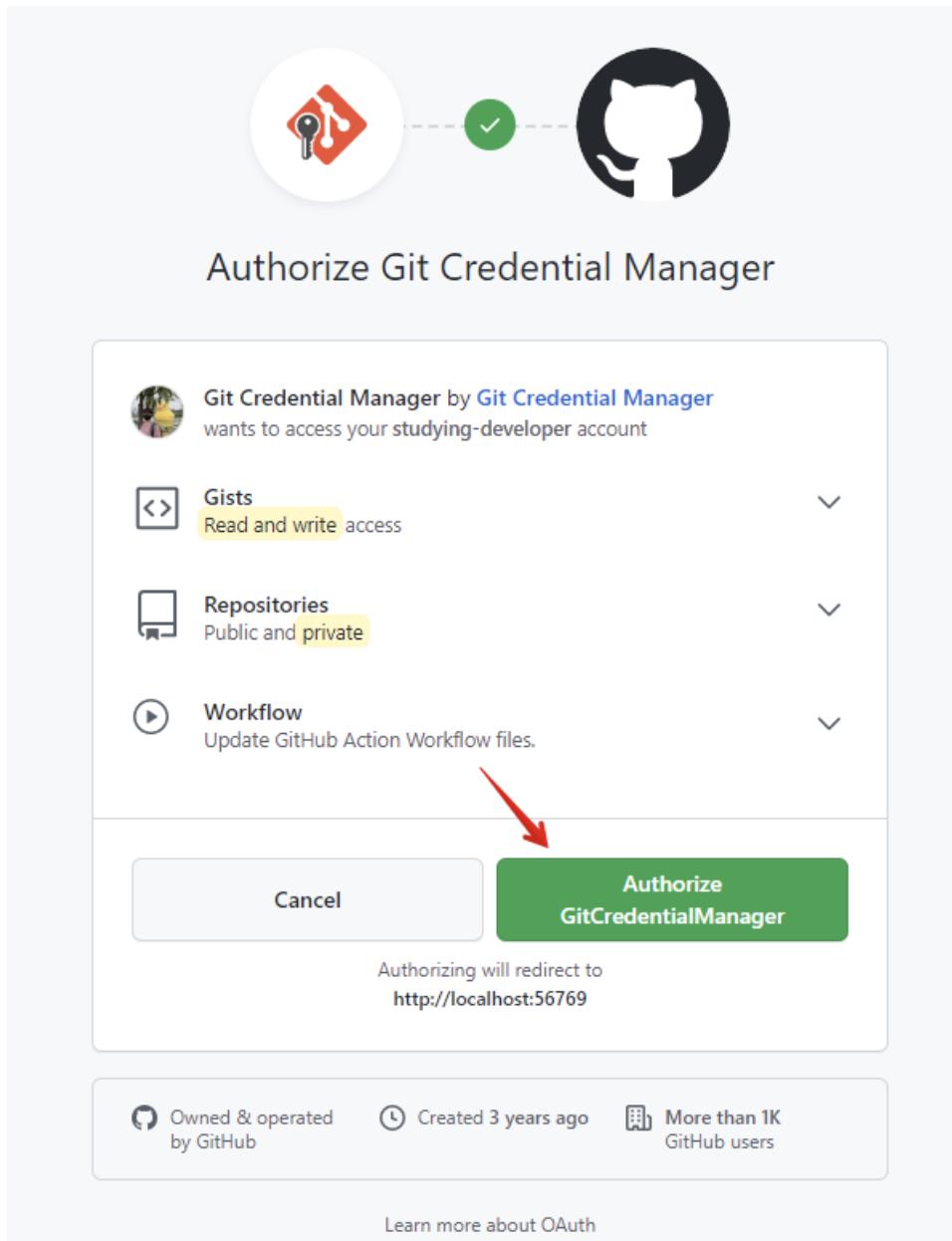
```
git config --global user.email "studyingdeveloper@gmail.com"  
git config --global user.name "studying-developer"
```

그런 다음 다시 commit을 해주기 위해 `git commit -m "첫 번째 커밋"` 이라고 다시 한번 타이핑 하자!

자 마지막으로는 우리가 송장을 붙인 택배를 보내야 한다! 이 명령어는 `git push`이다. 자 그런데, `git push` 를 타이핑하면 잘 안 되었다는 듯이 무언가 영어가 많이 나온다! 그 이유는 최초 1회에 한하여, `git push --set-upstream origin master` 이라는 명령어를 타이핑해 주어야 하기 때문이다!

이 명령어를 그대로 타이핑해 주자! 그러면 윈도우를 기준으로 최초 `git push` 를 하는 경우 아래 이미지가 나오게 된다! 이미지에서 “Sync in with your browser”를 클릭하면, 웹 사이트를 통해 권한 허가를 해주게 되고, 권한 허가를 해주면 `git push` 명령어가 성공해있다!!





그리고 github에 들어가 보면, 다음과 같이 우리의 코드가 원격에 저장된 것을 확인할 수 있다!! 🎉🎉🎉

studiying-developer / library-app Public

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Add file Code

**lannstarck 첫 번째 커밋** 0ca3768 7 minutes ago 1 commit

- gradle/wrapper 첫 번째 커밋 7 minutes ago
- src 첫 번째 커밋 7 minutes ago
- .gitignore 첫 번째 커밋 7 minutes ago
- build.gradle 첫 번째 커밋 7 minutes ago
- gradlew 첫 번째 커밋 7 minutes ago
- gradlew.bat 첫 번째 커밋 7 minutes ago
- settings.gradle 첫 번째 커밋 7 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

간단 도서관리 애플리케이션

0 stars 1 watching 0 forks

Releases

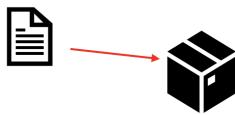
No releases published Create a new release

Packages

No packages published Publish your first package

Languages

Kotlin 94.6% HTML 4.7% Java 0.7%



코드를 택배 상자에 담기

git add .



택배에 송장 붙이기

git commit -m "적고 싶은 메시지"



택배 상자 github에 보내기

git push

매우 좋다~! 😊 이제 우리는 git과 github을 이용해 전용 컴퓨터에 코드를 보낼 준비도 마무리하였다. 이제 다음 시간에는 AWS에서 무료 전용 컴퓨터를 빌릴 것이다! 다음 시간에 AWS 가입 과정을 간단히 소개해 드릴 거지만,

## 41강. AWS의 EC2 사용하기

이번 시간에는 AWS 가입 과정을 간단히 안내드리고, AWS의 EC2라는 서비스를 이용하여 실제 전용 컴퓨터를 빌려볼 것이다.

AWS를 회원가입하기 위해서 아래 사이트에 들어가야 한다.

- <https://aws.amazon.com/ko/>

그 후, 화면 우측 상단에 있는 **AWS 계정 생성** 을 클릭하자. 아래 나오는 모든 스크린샷은 시간이 지남에 따라 AWS 사이트가 변경돼 조금 달라질 수 있다!



그럼 이제, 사이트에 회원가입을 하는 것처럼 이메일 주소와 계정 이름, 연락처 정보와 주소 등을 입력해야 한다.

## AWS에 가입

루트 사용자 이메일 주소  
계정 복구 및 일부 관리 기능에 사용

### AWS 계정 이름

계정의 이름을 선택합니다. 이름은 가입 후 계정 설정에서 변경할 수 있습니다.

**이메일 주소 확인**

# AWS에 가입

## 연락처 정보

AWS를 어떻게 사용할 계획이신가요?

- 비즈니스 – 업무, 학교 또는 조직의 경우
- 개인 – 자체 프로젝트의 경우

이 계정에 대해 누구에게 문의해야 하나요?

전체 이름

전화 번호

+1 222-333-4444

국가 또는 리전

미국

주소

아파트, 동, 호수, 빌딩, 층 등

시

시, 도 또는 리전

우편 번호

AWS 이용계약 [\[링크\]](#)을 읽었으며 이에 동의합니다.

**계속(2/5단계)**

특이하게 AWS 가입을 할 때에는 ‘결제 정보’도 입력을 해야 한다.

## 결제 정보

신용카드 번호



AWS는 현지에서 발급된 대부분의 신용카드를 허용합니다. 결제 옵션에 대해 자세히 알아보려면 [FAQ](#)를 참조하세요.

만료 날짜

▼ ▼

카드 소유자 이름

청구지 주소

- 내 연락처 주소 사용

- 새 주소 사용

이메일 주소

이메일 주소는 AWS와의 거래를 위해 VAT 영수증을 발송하는데 사용됩니다.

**확인 및 계속(3/5단계)**

확인 요금을 승인하기 위해 은행의 웹 사이트로 리디렉션될 수 있습니다.

가입 과정에서 입력한 카드는 실제 존재하는 카드인지 확인하기 위해 100원이 결제되었다가 취소된다. 그리고 AWS를 사용해 청구할 금액이 있다면, 이 카드로 자동 결제가 이루어진다!!

다음으로는 간단하게 휴대폰 인증 등을 진행하게 되고,

## 자격 증명 확인

AWS 계정을 사용하려면 먼저 전화번호를 확인해야 합니다. 계속하면 AWS 자동 시스템이 확인 코드 전송을 위해 연락합니다.

확인 코드를 어떻게 보내 드릴까요?

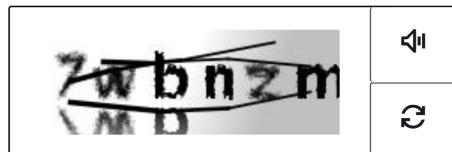
- 문자 메시지(SMS)
- 음성 통화

국가 또는 리전 코드

미국 (+1) ▾

휴대전화 번호

보안 검사



위에 보이는 문자를 입력하세요.

**SMS 전송(4/5단계)**

Support 플랜을 선택하게 된다. Support 플랜이란, AWS에 관련해 질문 같은 것을 AWS가 직접 대답해 줄 것인지, AWS 관련 기술 지원을 해주는 전문 인력을 적극 배치할 것인지 등을 선택하는 옵션이다.

우리는 현재 AWS를 처음 배우는 것이니 **기본 지원 - 무료**를 선택하면 된다.

## 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 레프 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



엔터프라이즈 수준의 지원이 필요하신가요?

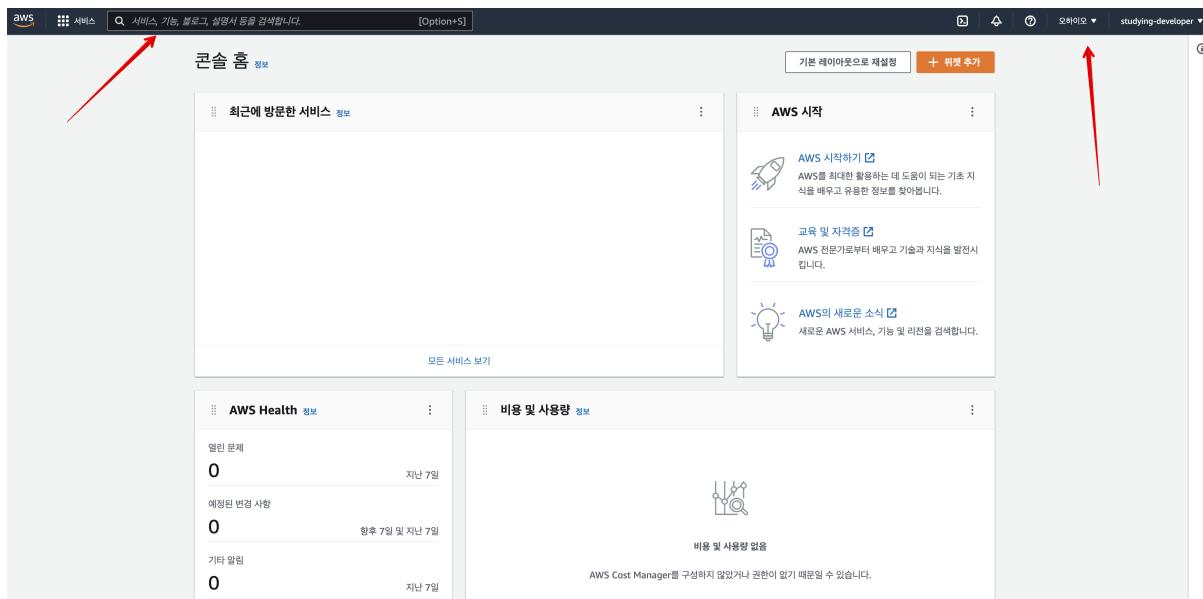


최저 월 15,000 USD로 15분 이내에 응답을 받을 수 있으며 기술 지원 관리자가 배정된 컨시어지 스타일의 서비스를 이용할 수 있습니다. [자세히 알아보기](#)

가입 완료

매우 좋다~!! 가입을 완료했다면, AWS 승인을 기다리게 되는 데 3~4시간이 걸릴 수도 있고 최대 하루 이틀이 걸릴 수도 있다.

AWS 가입 승인이 완료되었다면, 드디어 우리가 배포할 전용 컴퓨터를 빌릴 수 있다. 자 이제 AWS에 로그인해 보자! 로그인을 하게 되면 가장 먼저 다음과 같은 화면이 나오게 된다.



이 화면에서 오른쪽 화살표가 의미하는 것은 AWS의 지역이다. 우리가 AWS에서 컴퓨터를 빌릴 때 미국에 있는 컴퓨터를 빌릴 수도 있고, 한국에 있는 컴퓨터를 빌릴 수도 있다. 우리는 현재 한국에 있으니 한국 컴퓨터를 빌릴 수 있도록 지역을 변경해 주자.



다음으로 왼쪽 화살표에 있는 서비스에서 EC2를 검색하자. EC2는 Elastic Compute Cloud의 약자로, 탄력적으로 원격 컴퓨터를 사용할 수 있다는 의미이다.

EC2를 검색해 들어가면, 인스턴스 목록을 볼 수 있는 페이지가 있다. 왼쪽 side bar에서 인스턴스를 선택해도 되고, 대시보드 리소스에서 인스턴스를 클릭해도 된다. 인스턴스란, 우리가 빌린 컴퓨터를 의미한다.

The screenshot shows the AWS EC2 Dashboard. On the left, a sidebar menu is open under 'New EC2 Experience' with 'EC2 대시보드' selected. The main area displays '리소스' (Resources) with a summary for Asia Pacific (Seoul) Region. It shows 0 instances (running), 0 load balancers, 0 launch configurations, 2 security groups, 0 volumes, 1 instance type (t2.micro), 0 reserved instances, 0 snapshots, 0 spot instances, 0 dedicated hosts, 0 key pairs, and 0 elastic IPs. A note at the bottom says: 'AWS Launch Wizard for SQL Server를 사용하여 AWS에서 Microsoft SQL Server Always On 기용성 그룹을 손쉽게 크기 조정, 구성 및 배포할 수 있습니다. 자세히 알아보기'. Below this is the 'Instances' section with a heading '인스턴스 시작' (Launch Instance) and a note: '시작하려면 클라우드의 가상 서버인 Amazon EC2 인스턴스를 시작하십시오.' On the right, there's a '서비스 상태' (Service Status) section and a 'AWS Health 대시보드' link.

인스턴스 목록은 현재 당연히 비어 있다! 이제 오른쪽 상단의 인스턴스 시작을 눌러주자.

The screenshot shows the 'Instances' page. At the top, there are filters for '인스턴스 상태 = running' and a 'Find instances by attribute or tag (case-sensitive)' search bar. The main table has columns for 'Name', 'Instances ID', '인스턴스 상태' (Instance State), '인스턴스 유형' (Instance Type), '상태 검사' (Health Check), and '경보 상태' (Monitoring). A large orange button labeled '인스턴스 시작' (Launch Instance) is located at the top right of the table area. A red arrow points from the text above to this button.

인스턴스 시작을 누르면 다음과 같은 목록을 설정해 주어야 한다.

- 이름 및 태그
  - 우리가 빌릴 컴퓨터의 이름을 지정할 수 있다. 강의와 다르게 설정해도 무관한다.
- 애플리케이션 및 OS 이미지
  - 기본적으로 Amazon Linux 2 AMI가 설정되어 있다.
  - AWS에서 관리하는 리눅스 운영체제로, 딱히 수정할 필요가 없다.
- 인스턴스 유형
  - 우리가 빌릴 컴퓨터의 사양이다! CPU나 Memory처럼 H/W의 사양을 결정한다.

- `t2.micro` 를 선택하면 계정 생성 첫 1년 동안은 무료로 사용할 수 있다! `t2.micro` 를 선택해 주자.
- 인스턴스 유형은 `t2.micro .` 을 기준으로 나눠지는데 . 앞에 오는 문구의 의미는 컴퓨터의 성격과 세대를 의미하고 . 뒤에 오는 micro는 컴퓨터의 성능을 의미한다.
- 키 페어(로그인)



- 우리가 빌린 컴퓨터에 접속할 때 필요한 보안 파일(=키 페어)이다. 이 파일이 있어야만 우리가 빌린 컴퓨터에 접속할 수 있다.
- 현재는 계정을 처음 만들어 당연히 아무런 키 페어가 없으니, 새로운 키 페어를 만들어주자.

## 키 페어 생성

X

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

아래에 키 페어의 이름을 입력합니다. 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

키 페어 이름

키 페어 이름 입력

이름은 최대 255개의 ASCII 문자를 포함할 수 있습니다. 실행 또는 후행 공백은 포함할 수 없습니다.

키 페어 유형

RSA

RSA 암호화된 프라이빗 및 퍼블릭 키 페어

ED25519

ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식

.pem

OpenSSH와 함께 사용

.ppk

PuTTY와 함께 사용

취소

키 페어 생성

- 키 페어 유형 및 파일 형식은 변경할 필요 없이, 키 페어 이름만 적절하게 설정하면 된다!
- 그다음 새로 만든 키 페어로 설정해 주자.
- 네트워크 설정
  - 다른 것은 크게 설정할 필요가 없고, 보안 그룹을 새로 만들어 주자.

### 방화벽(보안 그룹) 정보

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

보안 그룹 생성

기존 보안 그룹 선택

다음 규칙을 사용하여 'launch-wizard-2'(이)라는 새 보안 그룹을 생성합니다.

에서 SSH 트래픽 허용  
인스턴스 연결에 도움

Anywhere  
0.0.0.0/0

▼

Allow HTTPS traffic from the internet  
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet  
To set up an endpoint, for example when creating a web server

- 보안 그룹의 역할은 우리가 빌린 컴퓨터에 접속할 때 어떤 IP를 허용할지, 어떤 포트로만 접근을 허용할지 결정하는 것이다.
- 3강에서 port란 프로그램이 통신할 때 사용하는 고유한 번호를 의미한다고 다루었다.
- 스토리지 구성
  - 우리가 빌릴 컴퓨터의 디스크 용량을 결정한다. 기본 8GB로 되어 있는데, 크게 변경할 내용은 없다.
- 고급 세부 정보
  - 여러 옵션을 설정할 수 있다. 지금은 딱히 설정해 줄 내용이 없으니, 넘어가도록 하자!

자 이렇게 이것저것 우리가 원하는 대로 설정을 해주고 **인스턴스 시작** 을 누르고 잠시 기다리면,

The screenshot shows the 'Create New Stack' configuration page for AWS CloudFormation. The configuration is set up for a single instance (1). The instance type is 't2.micro'. It uses the 'Amazon Linux 2 Kernel 5.10 AMI'. The security group is '새 보안 그룹'. A note indicates that the t2.micro instance type is limited to 750 hours of usage per month. The 'Storage' section shows a single volume of 8GiB. A red arrow points from the note about the t2.micro limitation to the '인스턴스 시작' (Launch Instance) button at the bottom right.

**▼ 요약**

인스턴스 개수 [정보](#)

1

**소프트웨어 이미지(AMI)**  
Amazon Linux 2 Kernel 5.10 AMI...[더 보기](#)  
ami-09cf633fe86e51bf0

**가상 서버 유형(인스턴스 유형)**  
t2.micro

**방화벽(보안 그룹)**  
새 보안 그룹

**스토리지(볼륨)**  
1개의 볼륨 - 8GiB

**프리 티어:** 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다. [X](#)

**취소** **인스턴스 시작**

드디어 인스턴스 목록을 보는 화면에서 실제 동작하는 컴퓨터를 볼 수 있다!!!

The screenshot shows the AWS CloudWatch Instances console. At the top, there are tabs for '인스턴스 (1) 정보' (Instance (1) Information), '연결' (Connect), '인스턴스 상태' (Instance Status), '작업' (Jobs), '인스턴스 시작' (Launch Instance), and a dropdown menu. Below the tabs is a search bar with the placeholder 'Find 인스턴스 by attribute or tag (case-sensitive)' and a filter button labeled '필터 지우기'. A filter is applied: '인스턴스 상태 = running'. The main table lists one instance: 'Library App Server' with ID 'i-01a01f13449d94919'. The status is '실행 중' (Running) with a green checkmark icon. The instance type is 't2.micro'. It has 2/2 successful health checks. There are buttons for '경보 설정' (Set Alarm) and '경보 없음' (No Alarms). Navigation icons for back, forward, and refresh are at the bottom.

너무 좋다~~!! 😊

이제 다음 Section에서는 실제로 우리가 빌린 컴퓨터에 접속해 배포를 해볼 것이다. 🔥🔥

## 42강. Section 6 정리. 다음으로!

이번 Section에서는 배포를 하기 위해 필요한 여러 준비들을 해보았다 👍 이 과정에서 다음과 같은 내용들을 배울 수 있었다!

1. 배포가 무엇인지 이해하고, 배포를 하기 위해 어떤 준비를 해야 하는지 알아본다.
2. 스프링 서버를 실행할 때 DB와 같은 설정들을 코드 변경 없이 제어할 수 있는 방법을 알아본다.
3. git과 github의 차이를 이해하고 git에 대한 기초적인 사용법을 알아본다.
4. AWS의 EC2가 무엇인지 이해하고, AWS를 통해 클라우드 컴퓨터를 빌려 본다.

자 이제 정말 거의 다 왔다~!! 이제 우리가 빌린 컴퓨터에 접속해 친구가 내가 만든 애플리케이션을 이용할 수 있도록 해보자!!! 🔥🏃