

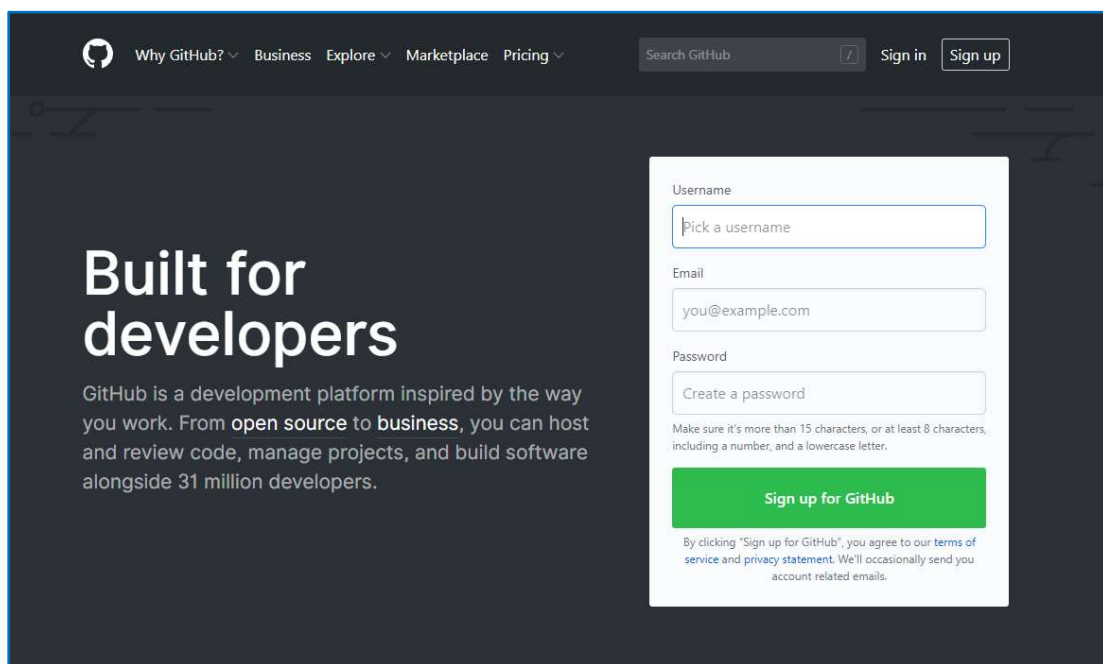
GitHub 소개 및 사용준비

1. What is GitHub?

- 1) Internet 어딘가의 (GitHub 이 관리하는) 원격 저장소
- 2) Google Drive 나 Dropbox 는 마지막으로 수정하고 업로드한 파일 한 개만 올라가지만, GitHub 에 우리의 Repository(Local 저장소)를 Push 하면, 지금까지 commit 한 모든 내역들 즉, Version 들의 내역들이 같이 Backup 이 되게 된다.
- 3) 따라서 지금까지 변경된 모든 사항들을 확인할 수 있다.

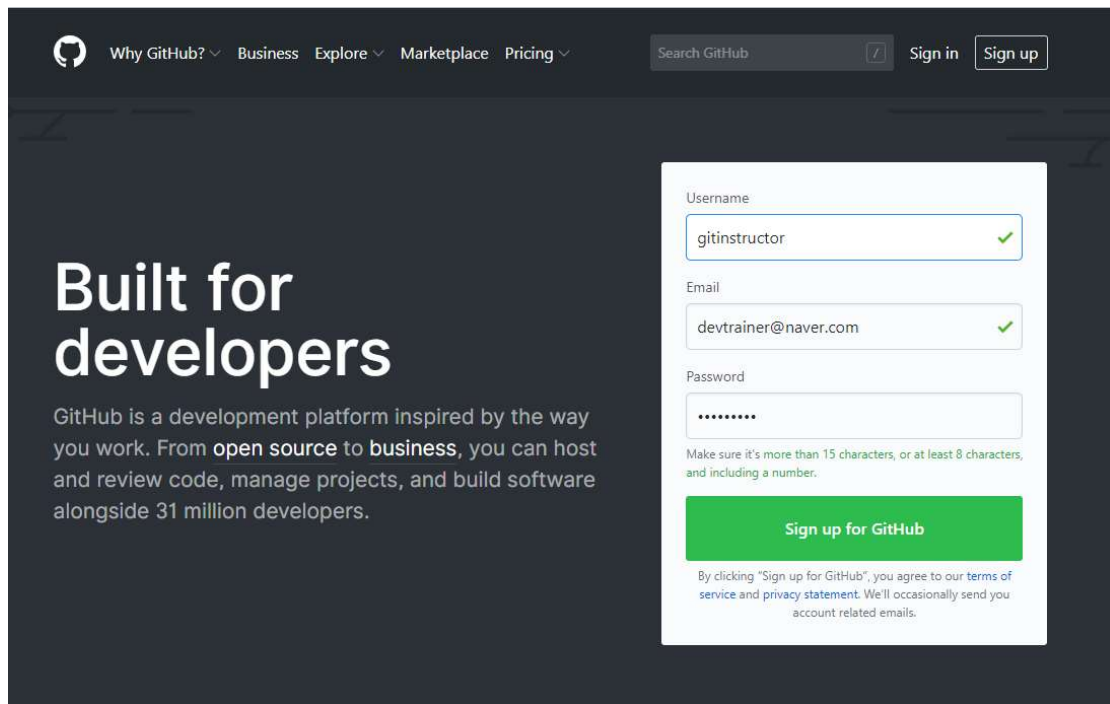
2. 계정생성

- 1) GitHub 을 사용하기 위해서 당연히 계정을 생성해야 한다.
- 2) 우선 Github homepage(<https://github.com/>)에 접속한다.



- 3) 'Username'에는 원하는 ID 를 영어로 입력한다.
- 4) 이때 입력하는 ID 는 '<http://github.com/yourID>'와 같은 형태로 생성될 것이다.

5) Email 과 Password 를 입력한다.



Why GitHub? Business Explore Marketplace Pricing Search GitHub Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 31 million developers.

Username: gitinstructor ✓

Email: devtrainer@naver.com ✓

Password:

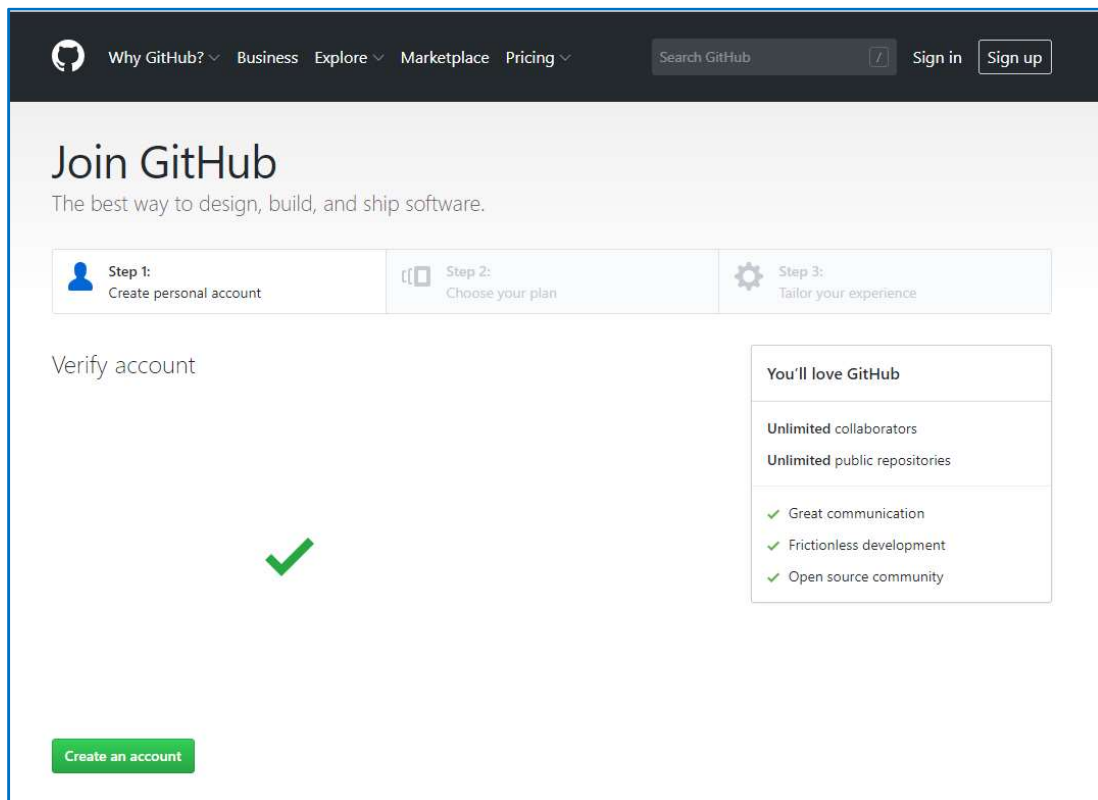
Make sure it's more than 15 characters, or at least 8 characters, and including a number.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

6) 모든 항목을 입력했으면 **[Sign up for GitHub]** 초록색 버튼을 눌러 다음을 진행한다.

7) 아래와 같이 [Step 1] 을 마치기 위해 **[Create an account]** 초록색 버튼을 눌러 다음을 진행한다.



Why GitHub? Business Explore Marketplace Pricing Search GitHub Sign in Sign up

Join GitHub

The best way to design, build, and ship software.

Step 1: Create personal account (active) | Step 2: Choose your plan | Step 3: Tailor your experience

Verify account

✓

You'll love GitHub

- Unlimited collaborators
- Unlimited public repositories
- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

Create an account

8) **[Welcome to GitHub]** 환영페이지가 나온다.

9) [Step 2]에서는 무료로 사용할 것인지, 월 7\$의 유료로 사용할 것인지를 결정한다.

The screenshot shows the GitHub 'Welcome to GitHub' page. At the top, there's a navigation bar with 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, a progress bar indicates three steps: 'Step 1: Completed Set up a personal account', 'Step 2: Choose your plan' (current step), and 'Step 3: Tailor your experience'. The main heading is 'Welcome to GitHub' with a subtitle 'You've taken your first step into a larger world, @gitinstructor.' Below this, the section 'Choose your personal plan' explains that every plan includes GitHub's most-loved features: Collaborative code review, issue tracking, the open source community, and the ability to join organizations. Two plans are presented: 'Free' (\$0 per month) and 'Developer' (\$7 per month). The 'Free' plan includes a personal account, unlimited public repositories, and unlimited collaborators. The 'Developer' plan includes a personal account, unlimited public repositories, unlimited private repositories, and unlimited collaborators. It also mentions that it's free for students as part of the Student Developer Pack. At the bottom, there are two checkboxes: 'Help me set up an organization next' and 'Send me updates on GitHub news, offers, and events'. A green 'Continue' button is at the bottom right.

10) 전문적으로 사용할 것이 아니면 무료로 사용할 것을 권장한다.

11) 향후 보다 더 전문적인 작업을 할 때 그때 유료로 전환하면 된다.

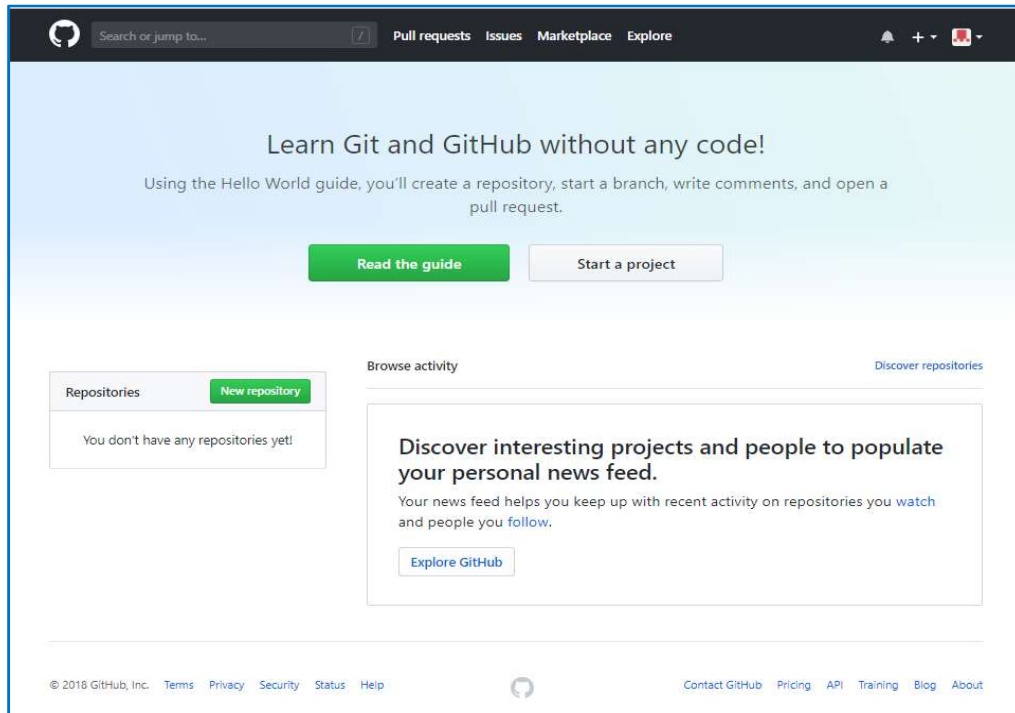
12) [Continue] 초록색 버튼을 눌러 다음을 진행한다.

The screenshot shows the GitHub 'Welcome to GitHub' page, specifically the 'Tailor your experience' step. The progress bar at the top shows 'Step 1: Completed Set up a personal account', 'Step 2: Choose your plan', and 'Step 3: Tailor your experience' (current step). The main heading is 'Welcome to GitHub' with a subtitle 'You'll find endless opportunities to learn, code, and create, @gitinstructor.' Below this, the section 'How would you describe your level of programming experience?' has three radio button options: 'Very experienced', 'Somewhat experienced' (selected), and 'Totally new to programming'. The next section, 'What do you plan to use GitHub for? (check all that apply)', has six checkboxes: 'Development', 'Research', 'School projects', 'Project Management', 'Design', and 'Other (please specify)'. The 'Development' checkbox is selected. The third section, 'Which is closest to how you would describe yourself?', has three radio button options: 'I'm a student', 'I'm a professional' (selected), and 'I'm a hobbyist'. The fourth section, 'What are you interested in?', has a text input field with a list of tags: 'machine-learning', 'data-science', 'development', 'something-holic', and 'teaching'. At the bottom, there are two buttons: a green 'Submit' button and a 'skip this step' link.

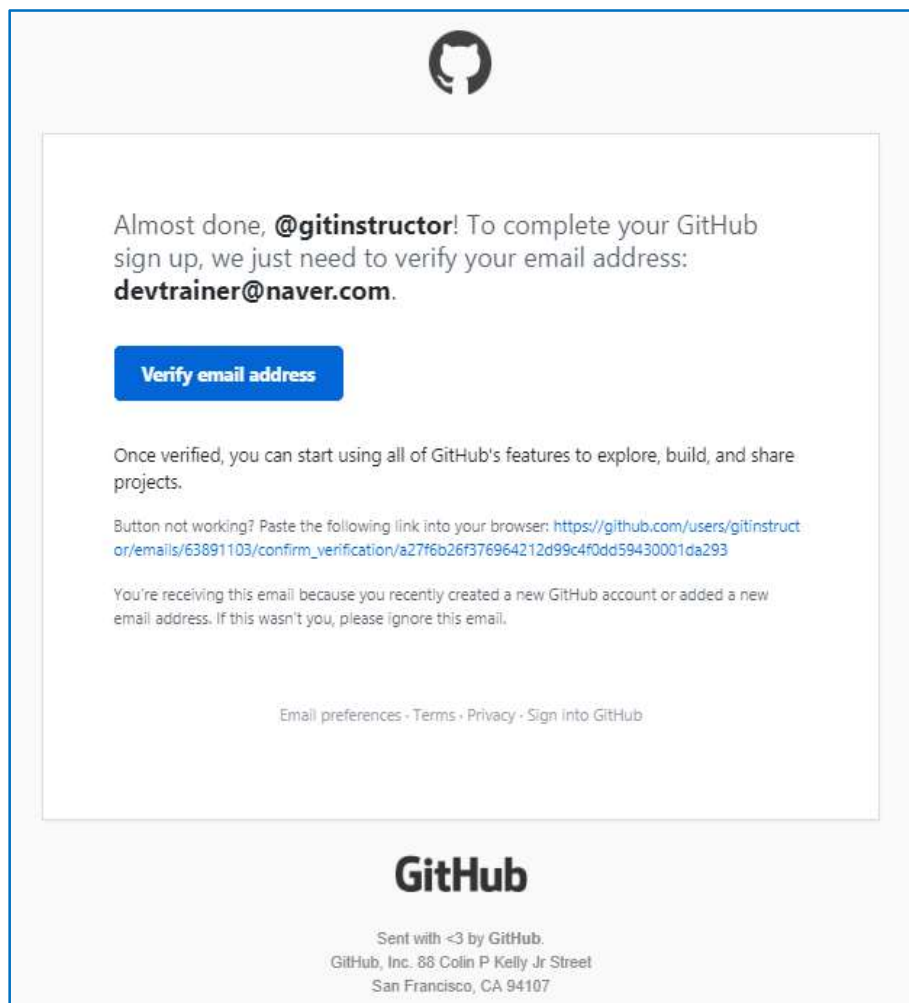
13) [Step 3] 단계이다.

14) 굳이 이 페이지의 각 항목들을 체크할 필요는 없다.

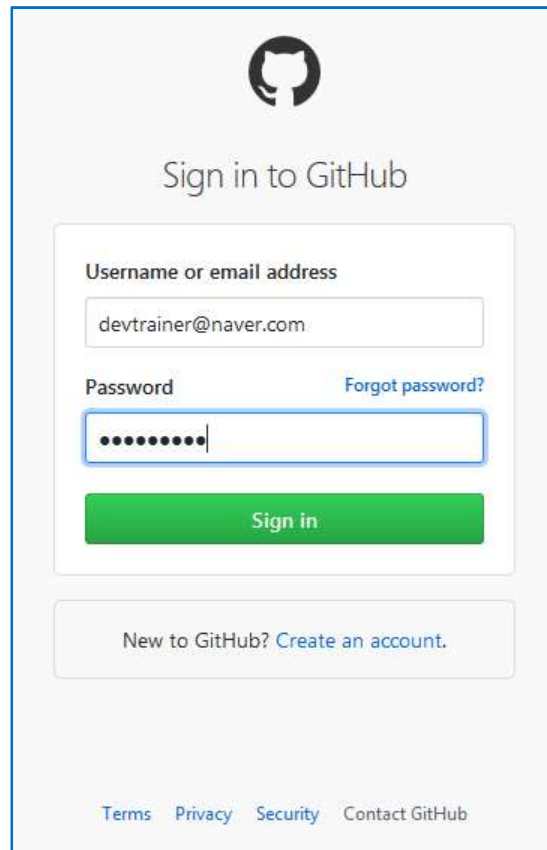
15) 필요한 항목을 입력했다면 [Submit] 초록색 버튼을 눌러 가입절차를 완성한다.



16) GitHub 에서 계정 가입 절차가 모두 끝나면 등록한 Email 로 검증메일이 날라온다.

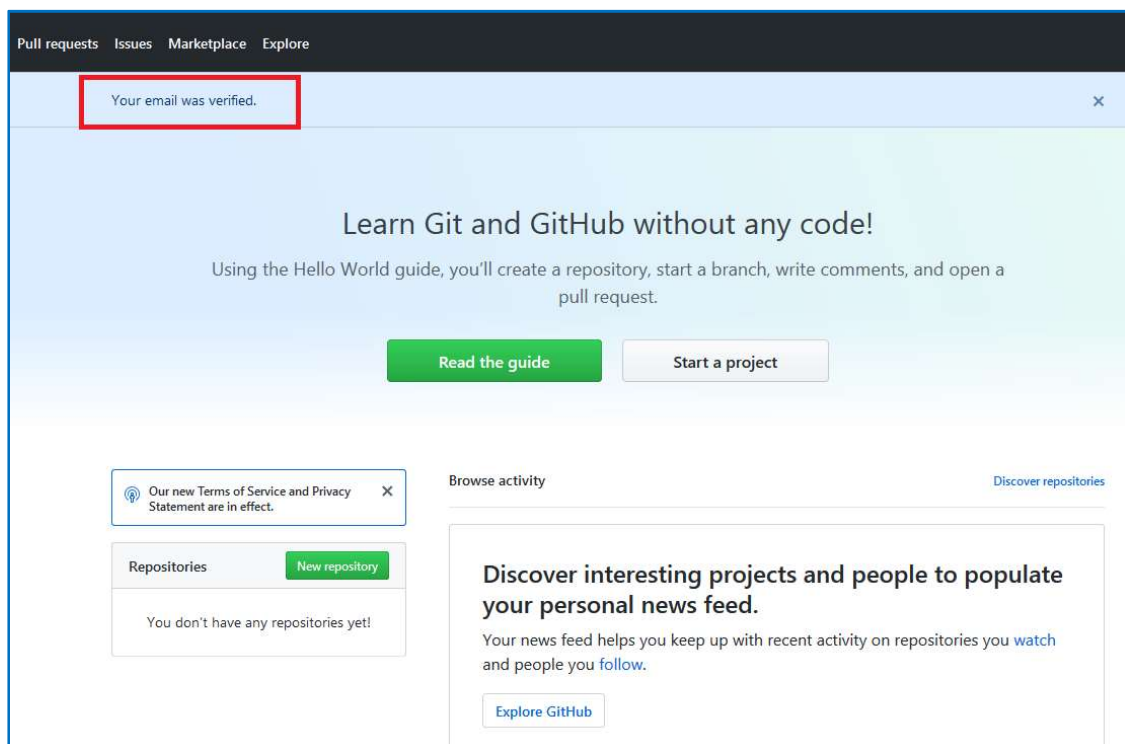


17) [Verify email address]의 파란색 Button 을 누르면 된다.



The image shows the GitHub sign-in page. At the top is the GitHub logo and the text "Sign in to GitHub". Below this is a form with two input fields: "Username or email address" containing "devtrainer@naver.com" and "Password" with masked characters. A blue link "Forgot password?" is next to the password field. A green "Sign in" button is below the fields. At the bottom of the form is a link "New to GitHub? Create an account.". At the very bottom of the page are links for "Terms", "Privacy", "Security", and "Contact GitHub".

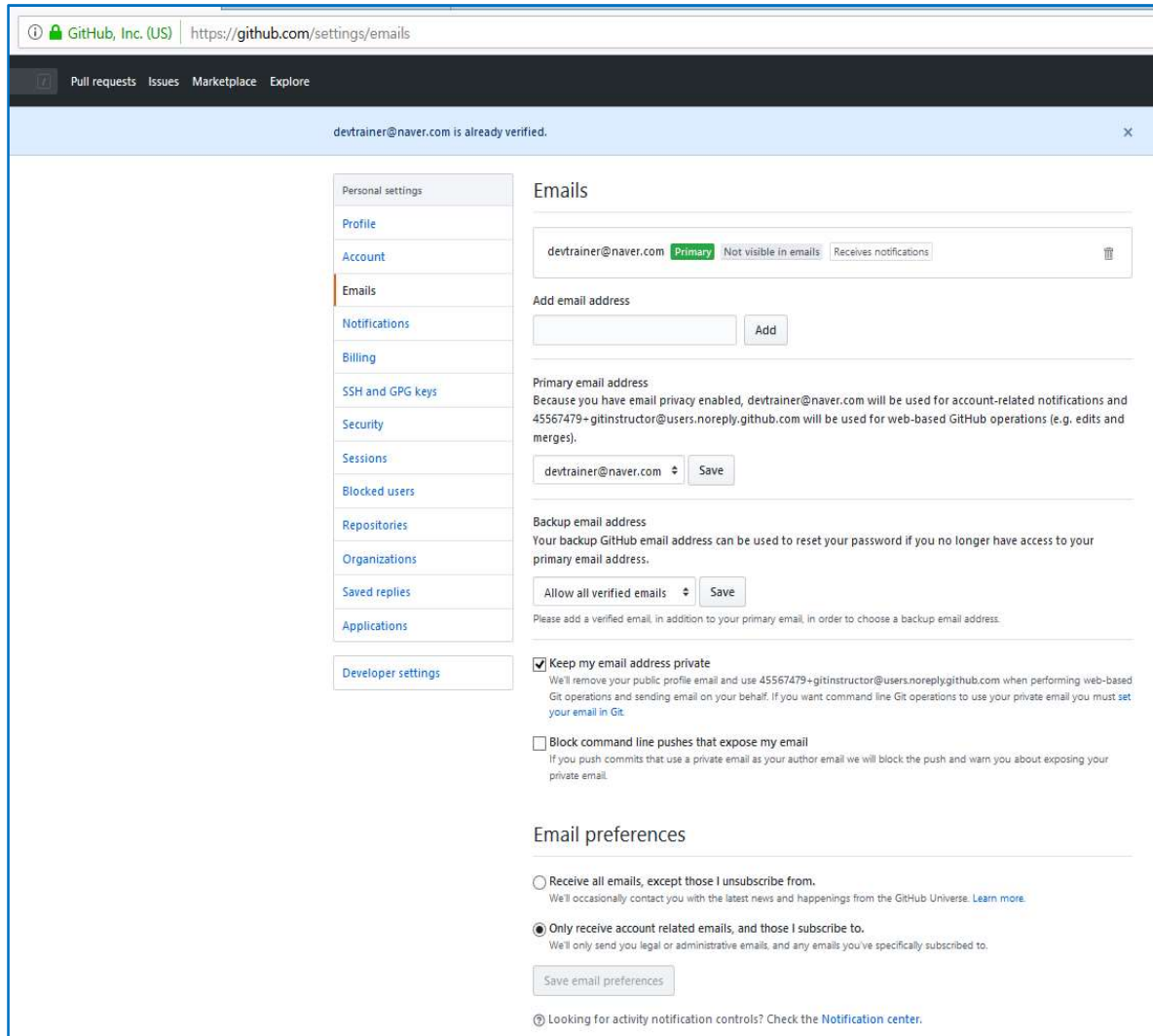
18) GitHub 에 login 한다.



19) Page 상단에 보면 [Your email was verified.]라는 문구가 보인다.

20) 정상적으로 Email 검증되었다.

21) 혹시 받은 Email 에서 **[Verify email address]** Button 을 Click 하지 않고 그 아래에 있는 Link 를 Click 하면 다음과 같은 화면으로 넘어간다.



22) 어쨌든 Email 검증이 완료되었다.

3. SSH Key 설정 - Deprecated

- 1) GitHub 은 작성한 repository 에 대한 접근 인증을 SSH 공개 key 로 한다.
- 2) 공개 key 인증에 필요한 SSH 공개 key 를 설정하고 GitHub 에 등록하기로 한다.
- 3) 다음과 같이 실행해서 SSH Key 를 작성한다.

```
$ ssh-keygen -t rsa -C "devtrainer@naver.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Instructor/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
$ ssh-keygen -t rsa -C <Your-Email Address>"
```

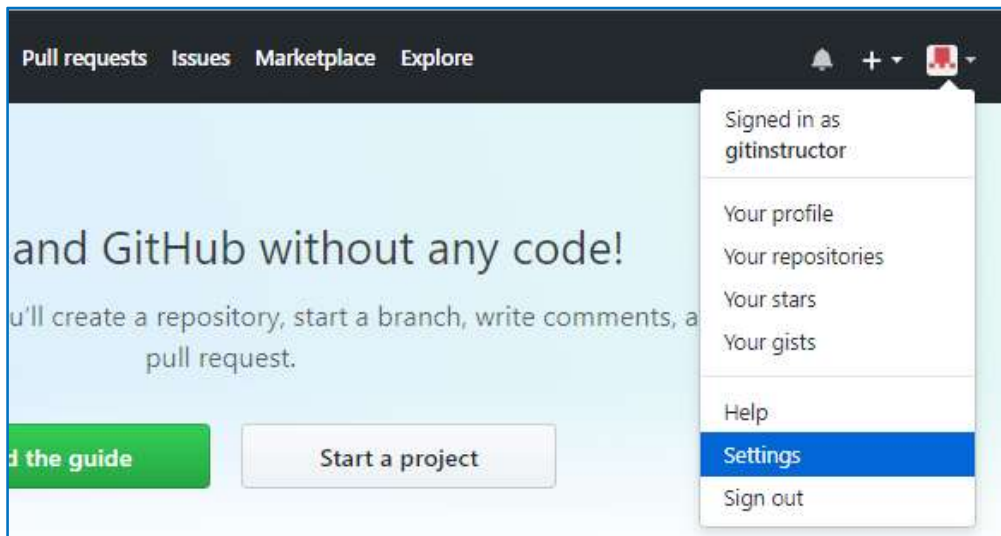
- 4) 즉 GitHub 에 등록할 때 사용한 자신의 Email 주소를 넣는다.
- 5) 'Enter file in which to save the key' 항목은 저장할 위치를 지정하는 것인데, 그냥 Enter Key 를 넣는다.
- 6) 비밀번호는 인증할 때 입력하는데, 외우기 쉬우면서도 복잡한 비밀번호를 넣는다.
- 7) 비밀번호를 입력하고 나면 다음과 같이 출력된다.

```
Your identification has been saved in /c/Users/Instructor/.ssh/id_rsa.
Your public key has been saved in /c/Users/Instructor/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256: [REDACTED] devtrainer@naver.com
The key's randomart image is:
+---[RSA 2048]---+
|                 .                 |
|                . . o              |
|               + .o                |
|              o o                  |
|             o .S   o .            |
|            o ...E  .o+ o          |
|           =o=   . o.o==           |
|          .B.Oo.  o ooB=           |
|         o+* +o.. .oBo=           |
+-----[SHA256]-----+
```

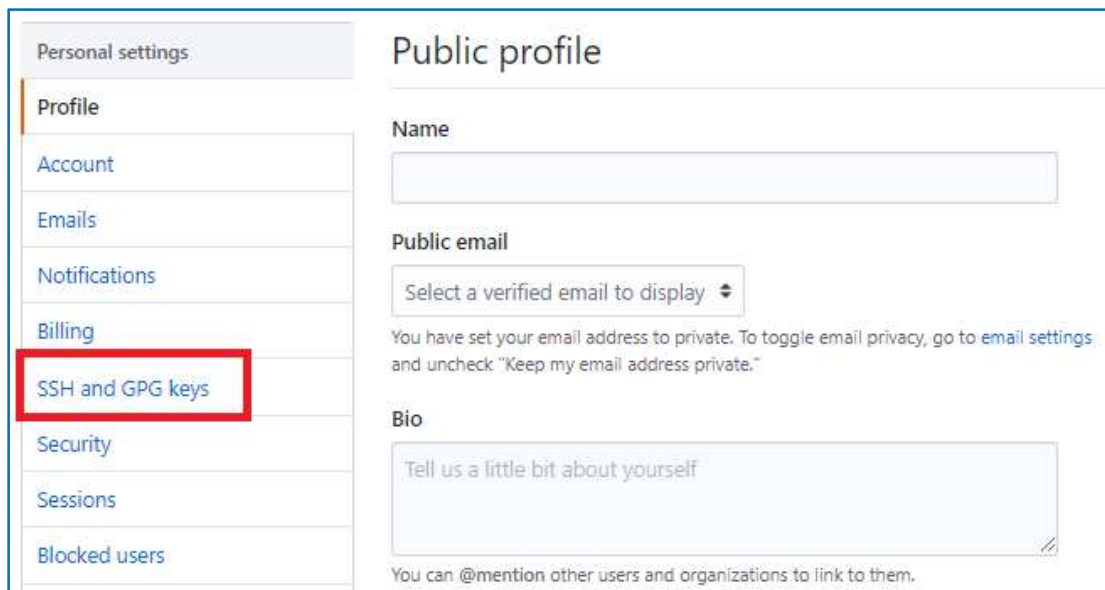
- 8) id_rsa 는 비밀 key file 이고, id_rsa.pub 는 공개 key file 이다.

4. 공개 key 등록 - Deprecated

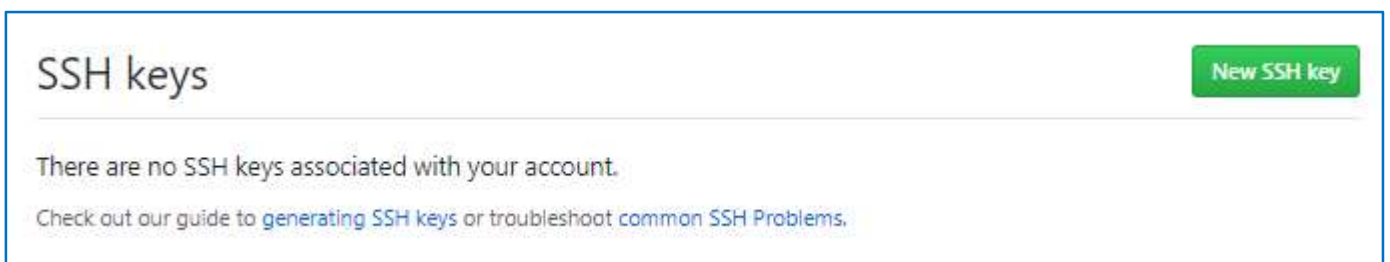
- 1) GitHub 에 공개 key 를 등록하고, 비밀 key 를 사용해서 GitHub 에 인증해 보자.
- 2) 오른쪽 상단의 계정을 클릭하면 drop-down menu 가 나오는데, 거기서 **[Settings]**를 click 한다.



- 3) 다음과 같은 설정 창에서 **[Account settings]** > **[SSH and GPG keys]** menu 를 click 한다.



- 4) 다음과 같은 창에서 초록색 **[New SSH key]** Button 을 Click 한다.



- 5) **[Title]**에는 해당 key 의 이름을 입력하고, **[Key]**에는 id_rsa.pub 의 내용을 복사해서 붙여 넣는다.

6) id_rsa.pub 의 내용을 보려면 해당 key 가 있는 곳으로 이동하여 아래와 같이 확인한다.

```
Instructor@DESKTOP-NU7GQVV MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts

Instructor@DESKTOP-NU7GQVV MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa [REDACTED] devtrainer@naver.co
m
```

7) id_rsa.pub 의 내용을 복사해서 넣고 초록색 **[Add SSH key]** Button 을 Click 한다.

SSH keys / Add new

Title

GitHub for Windows-gitinstructor

Key

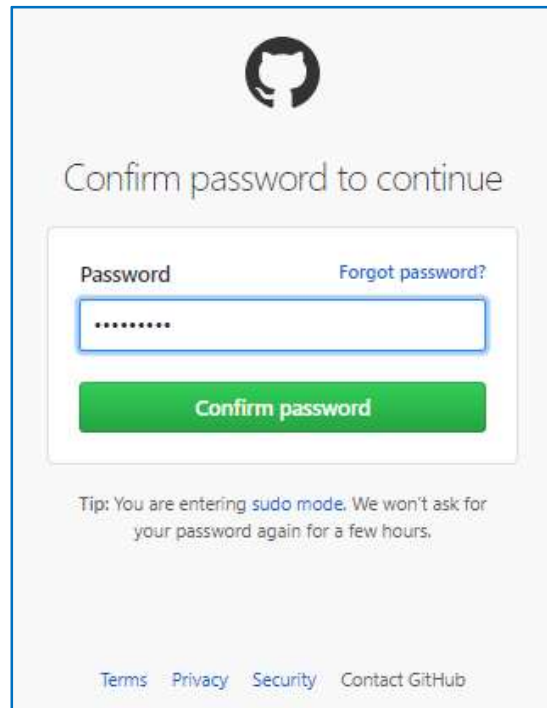
ssh-rsa

[REDACTED]

devtrainer@naver.com

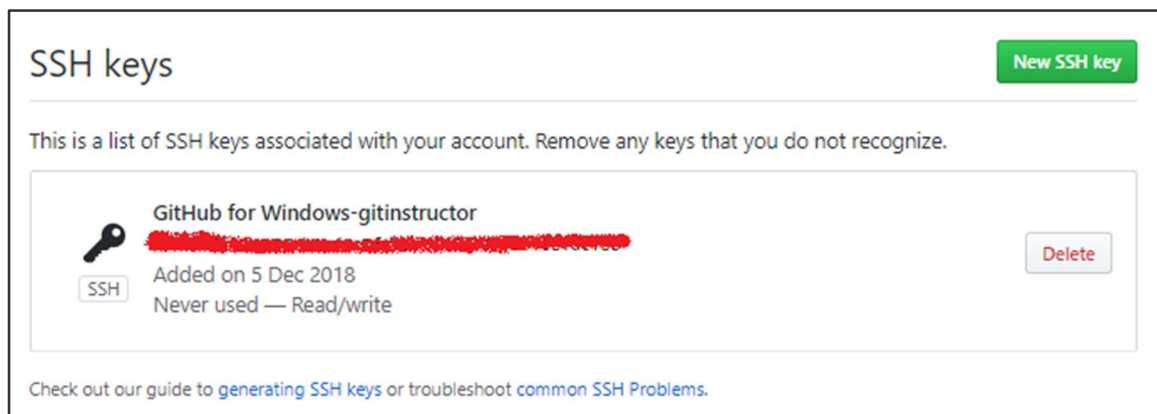
Add SSH key

8) 본인 여부를 검증하기 위해 계정에 대한 비밀번호를 요구할 수 있다.



The image shows a GitHub password confirmation screen. At the top is the GitHub logo. Below it, the text "Confirm password to continue" is displayed. There is a "Password" label and a "Forgot password?" link. A password input field contains several dots. Below the input field is a green button labeled "Confirm password". At the bottom, there is a tip: "Tip: You are entering sudo mode. We won't ask for your password again for a few hours." and links for "Terms", "Privacy", "Security", and "Contact GitHub".

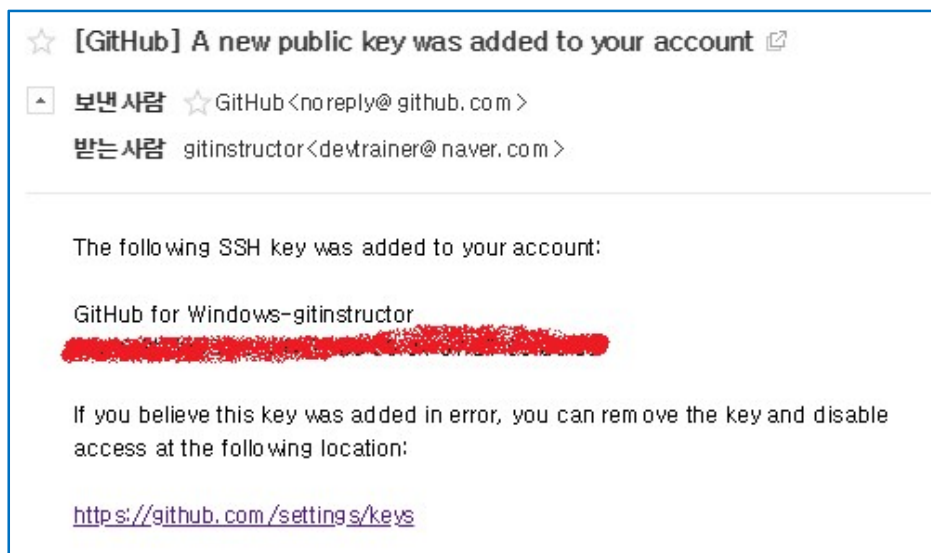
9) 가입시 입력했던 비밀번호를 넣고 **[Confirm password]** 초록색 Button 을 Click 한다.



The image shows the "SSH keys" page on GitHub. At the top right is a green button labeled "New SSH key". Below the title, there is a message: "This is a list of SSH keys associated with your account. Remove any keys that you do not recognize." There is a list of SSH keys. The first key is "GitHub for Windows-gitinstructor" with a redacted public key. It was added on 5 Dec 2018 and has never been used. There is a "Delete" button next to it. At the bottom, there is a link to "Check out our guide to generating SSH keys or troubleshoot common SSH Problems."

10) 등록이 무사히 끝나면 공개 Key 등록 완료 Page 를 확인할 수 있다.

11) 또한 가입시 등록했던 Mail 주소로 Email 이 날라온다.



The image shows an email notification from GitHub. The subject is "[GitHub] A new public key was added to your account". The sender is "보낸 사람 GitHub <noreply@github.com>". The recipient is "받는 사람 gitinstructor <devtrainer@naver.com>". The body of the email says: "The following SSH key was added to your account: GitHub for Windows-gitinstructor" followed by a redacted public key. It then says: "If you believe this key was added in error, you can remove the key and disable access at the following location: [http://github.com/settings/keys](\"http://github.com/settings/keys\")".

12) 지금까지의 설정이 모두 끝났다면 만들어진 비밀 Key 로 GitHub 에 인증하거나 통신할 수 있다.

13) 실제로 동작하는지 확인해 보자.

14) 다음을 입력한다.

```
$ ssh -T git@github.com
```

```
$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.255.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.255.113' (RSA) to the list of known hosts.
Enter passphrase for key '/c/Users/Instructor/.ssh/id_rsa':
```

15) [Are **you sure you want to continue connecting (yes/no)?**]에서 **yes** 를 입력한다.

16) 다시 한번 SSH key 를 생성할 때 넣었던 비밀번호를 입력한다.

17) 다음과 같이 출력되면 성공한 것이다.

```
Hi gitinstructor! You've successfully authenticated, but GitHub does not provide shell access.
```

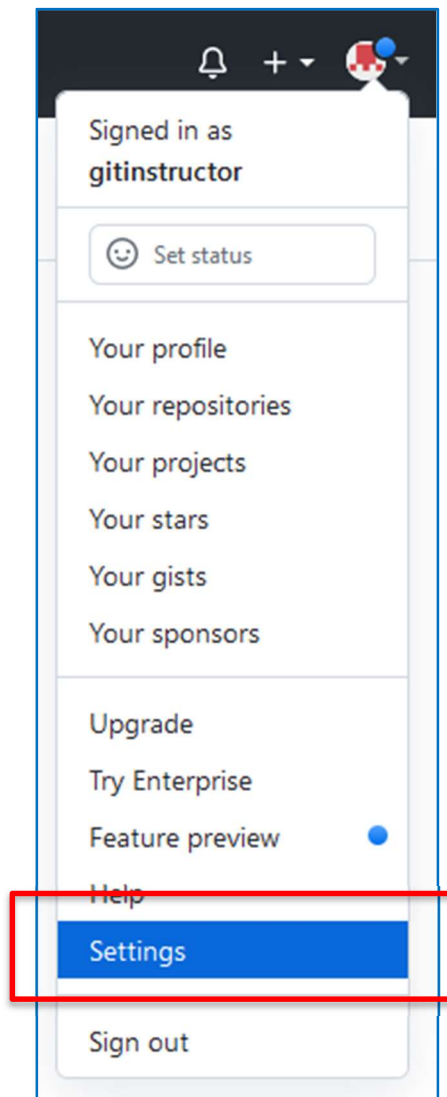
5. GitHub Personal Access Token 만들기

- 1) GitHub 은 2021 년 8 월 13 일 기준으로 더 이상 Password 방식의 인증을 지원하지 않게 되었다.

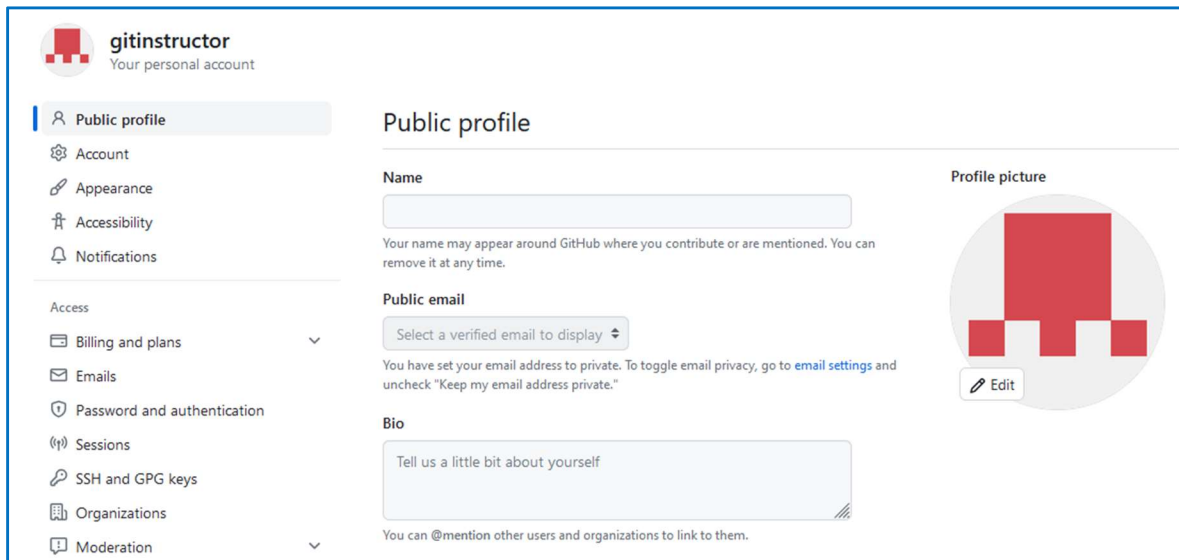
```
→ gotetris git:(main) git push -u origin main
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: unable to access 'https://github.com/cpro95/gotetris.git/': The requested URL returned error: 403
```

[<https://cpro95.tistory.com/456>]

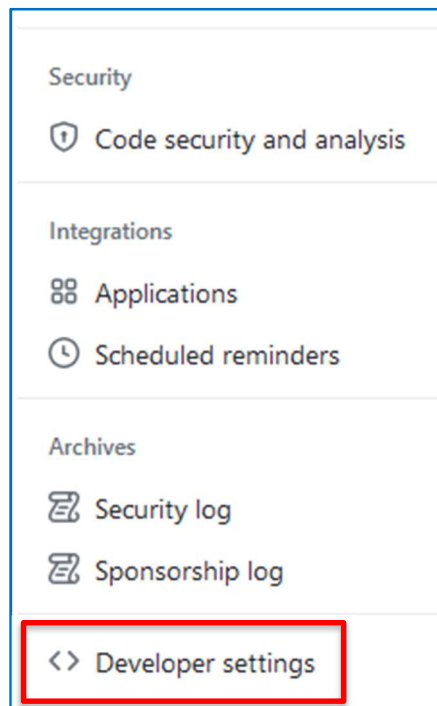
- 2) Refer to <https://docs.github.com/ko/enterprise-server@3.5/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>
- 3) 명령줄 또는 API 를 사용하여 GitHub 에 인증할 때 암호 대신 Personal Access Token 을 사용한다.
- 4) 먼저, GitHub 에 로그인 후, 페이지의 우측 상단에서, 계정 > [Settings]를 클릭한다.



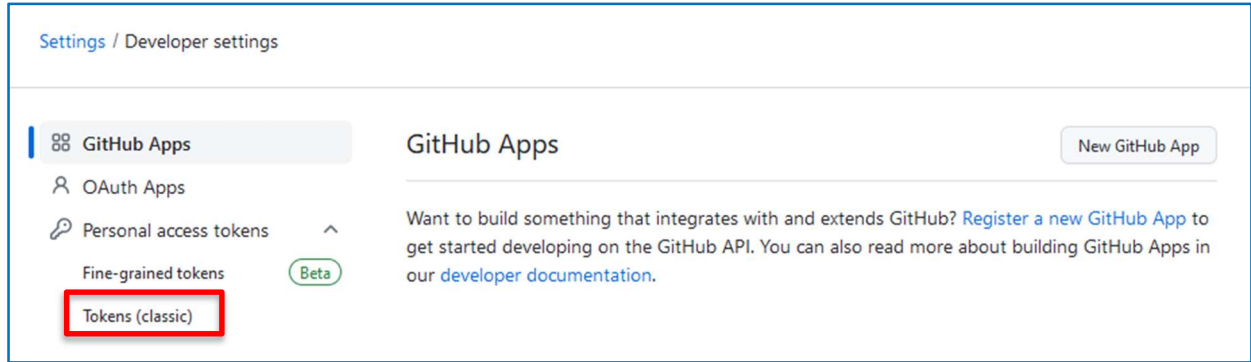
5) GitHub 계정의 상세 페이지로 이동한다.



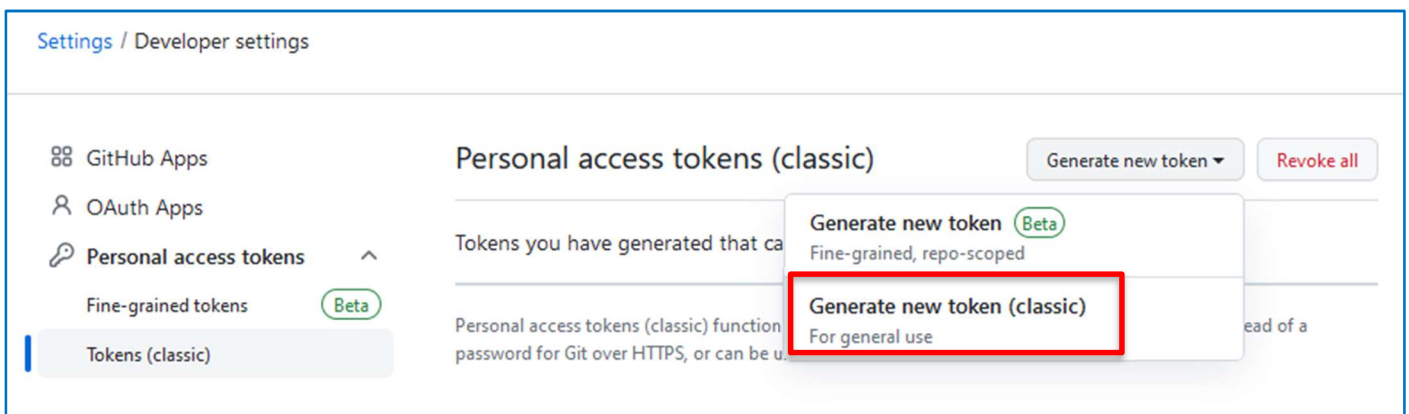
6) 페이지 좌측 메뉴의 제일 아래에 있는 [Developer settings] 메뉴를 클릭한다.



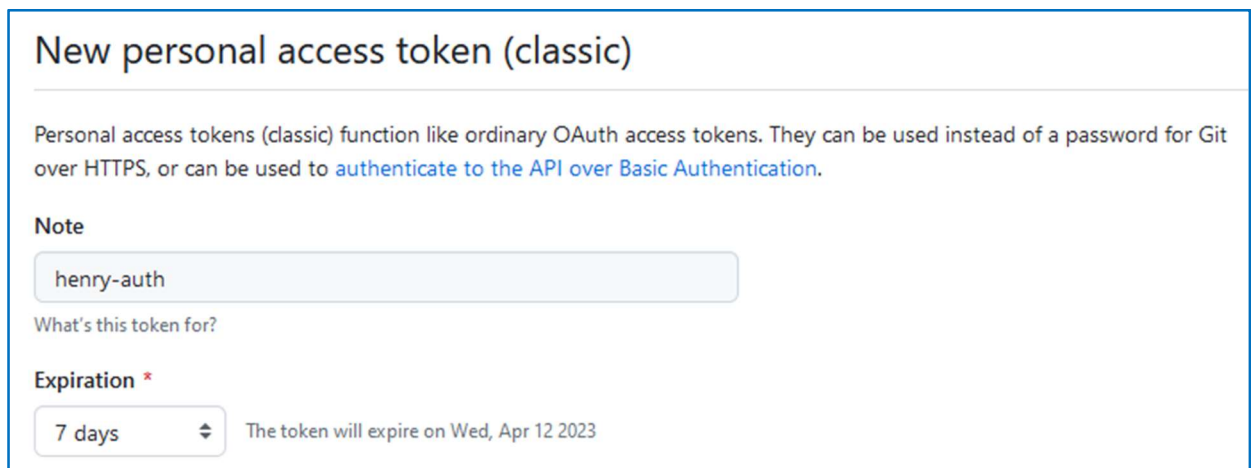
- 7) [Developer settings] 페이지에서 좌측 메뉴의 [Personal access tokens] > [Tokens(classic)]를 클릭한다.



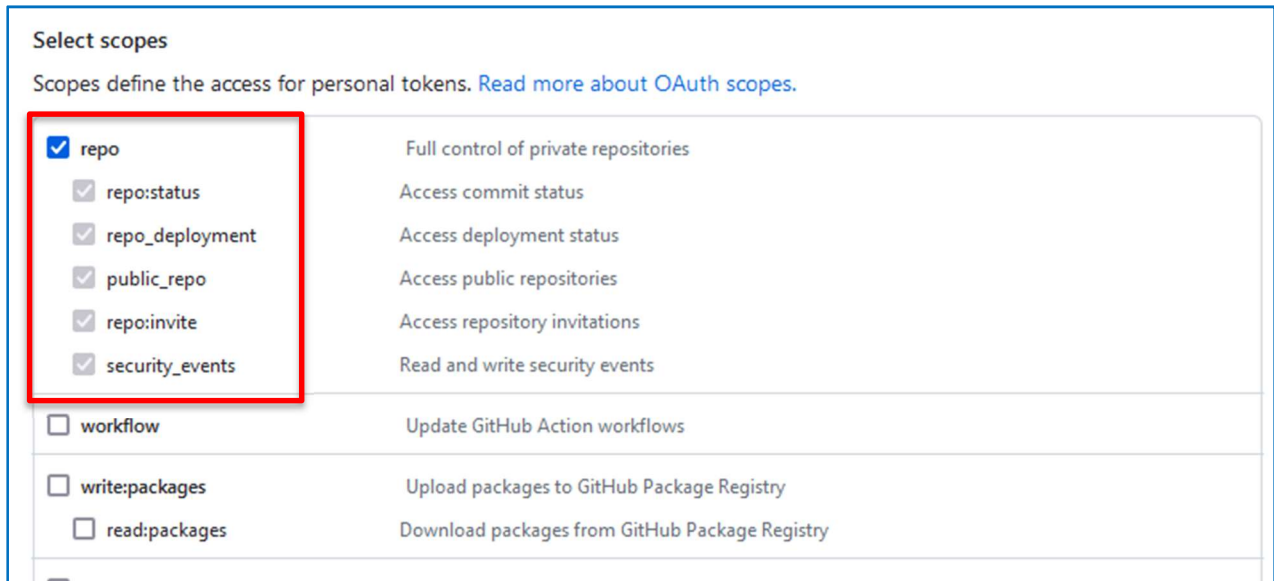
- 8) [Generate new token] > [Generate new token(classic)] 버튼을 클릭한다.



- 9) Token 이름을 넣고, Token 의 만료기간을 설정한다. 만료기간 드롭다운 메뉴를 클릭하여 기간을 설정할 수 있다.



10) 지금 생성하는 Token 에게 부여할 범위를 선택한다. Token 을 사용해서 명령줄에서 Repository 에 접근하려면 repo 를 선택한다. 할당된 범위가 없는 Token 은 Public 정보에만 접근할 수 있다. 기본적인 repo 를 체크한다.

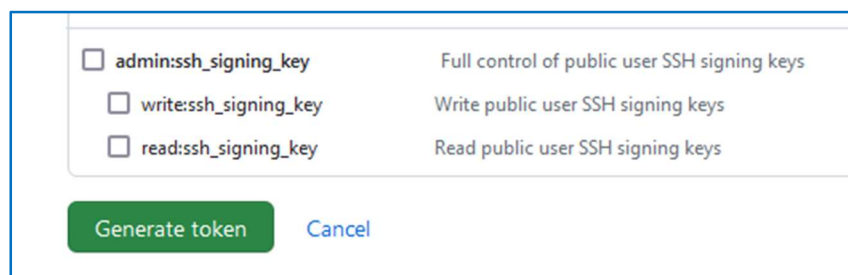


Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry

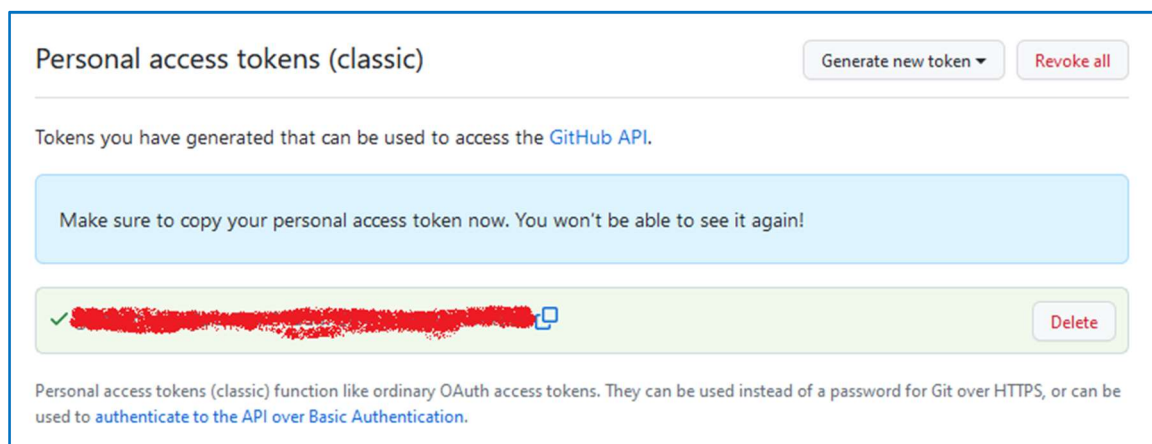
11) 범위의 선택이 끝나면 페이지 제일 밑으로 내려와서 [Generate token] 초록색 버튼을 클릭한다.



<input type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

[Generate token](#) [Cancel](#)

12) Token 이 생성되었다. 다음 그림과 같이 생성된 Token 값을 확인하고, 메모장 같은 편집기에 복사해서 붙여 넣은 다음 안전한 곳에 저장한다. 지금 외에는 생성된 Token 값을 확인할 수 없기 때문에 반드시 값을 복사해서 파일에 저장한다.



Personal access tokens (classic) [Generate new token](#) [Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ [REDACTED] [Copy](#) [Delete](#)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

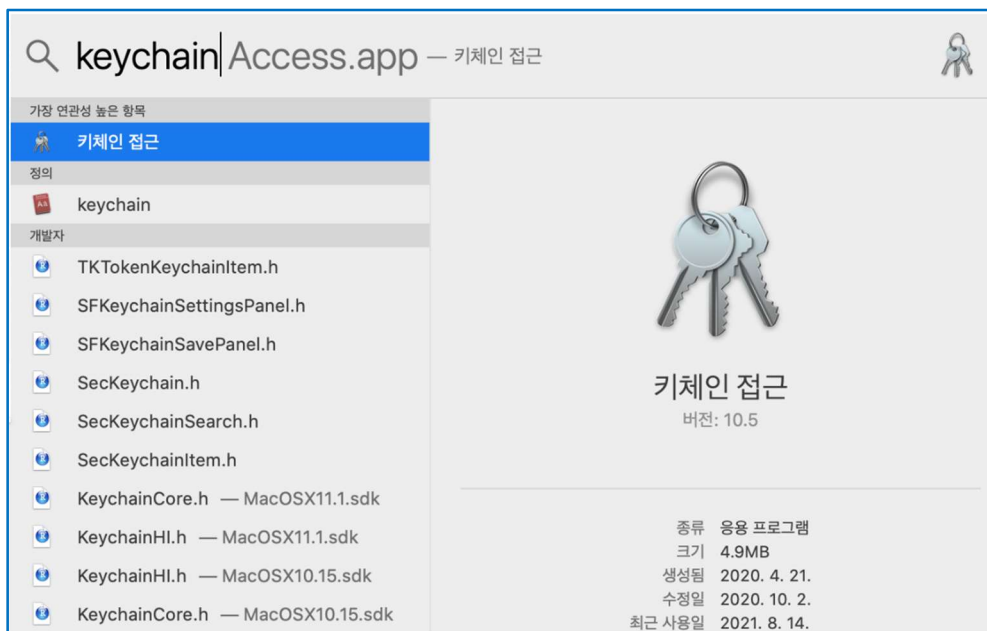
13) 지금부터 GitHub 에 Push 같은 작업을 수행할 때, 패스워드를 입력하라고 하면, 방금 생성한 Access Token 값을 입력하면 된다.

14) macOS 에서 Push 할 때, 아래의 그림과 같은 로그가 보이며, 안될 수 있다.

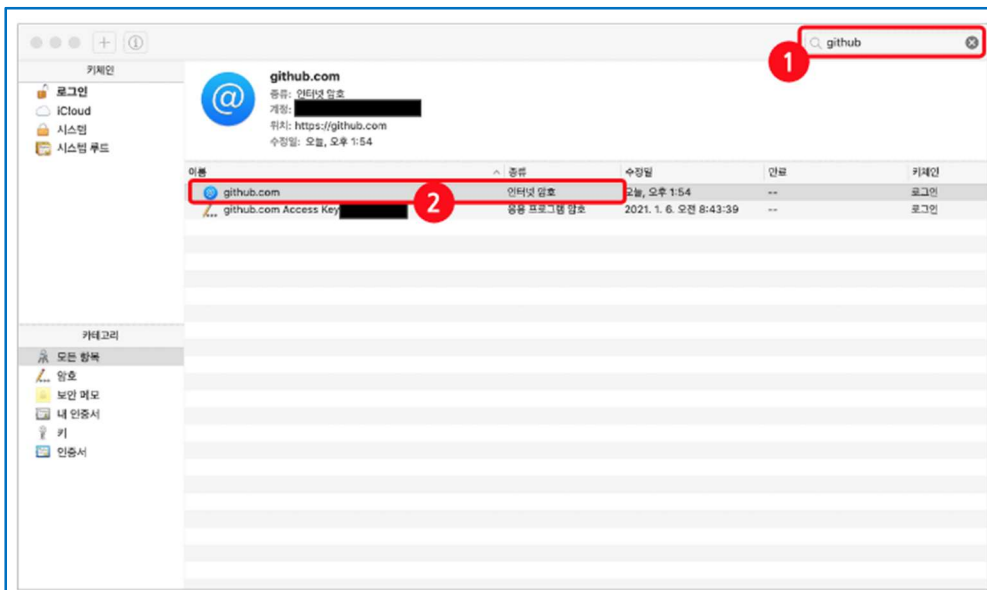
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead. ... The requested URL returned error: 403

15) 이럴 때는 KeyChain 에 새로 등록하면 되는데, 발생원인은 다른 값이 이미 KeyChain 에 등록되어 있어서 서로 값이 다르기 때문에 발생한다. 변경하는 방법은 다음 사이트를 참고하였다. <https://curryou.tistory.com/403>

- Spotlight 검색에서 keychain 을 찾아서 실행



- "github" 검색 후, 종류가 "인터넷암호"인 항목을 찾아서 더블 클릭

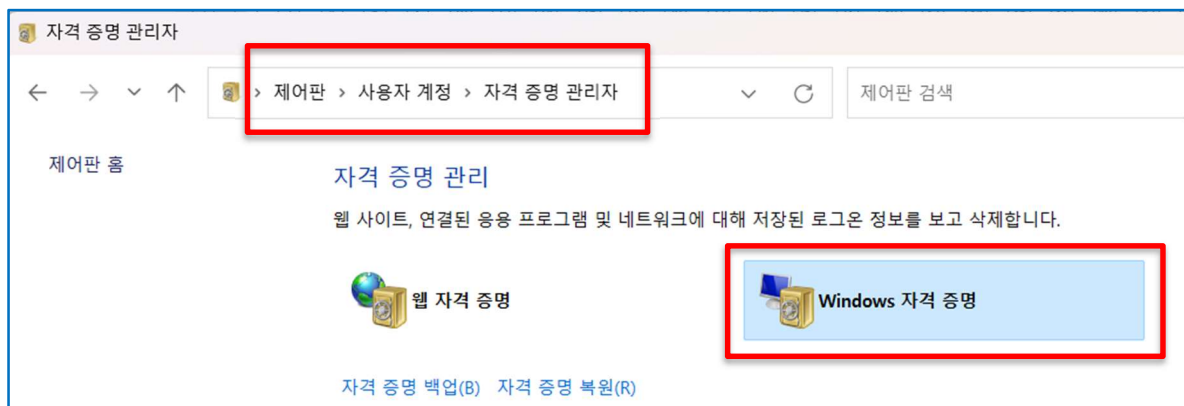


- 암호보기 항목을 체크 후, 기존의 패스워드를 발급 받은 Access Token 값으로 변경

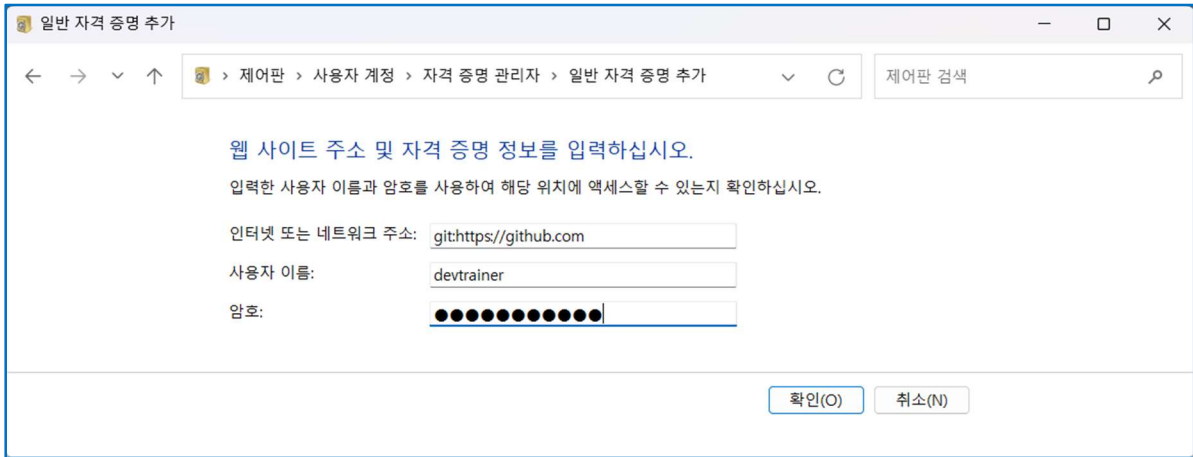
16) 오류가 발생하지 않더라도, 새로 macOS 에 GitHub Access Token 을 등록하는 방법도 같다.

17) 다음은 Windows 에 GitHub Token 을 설정하는 방법이다.

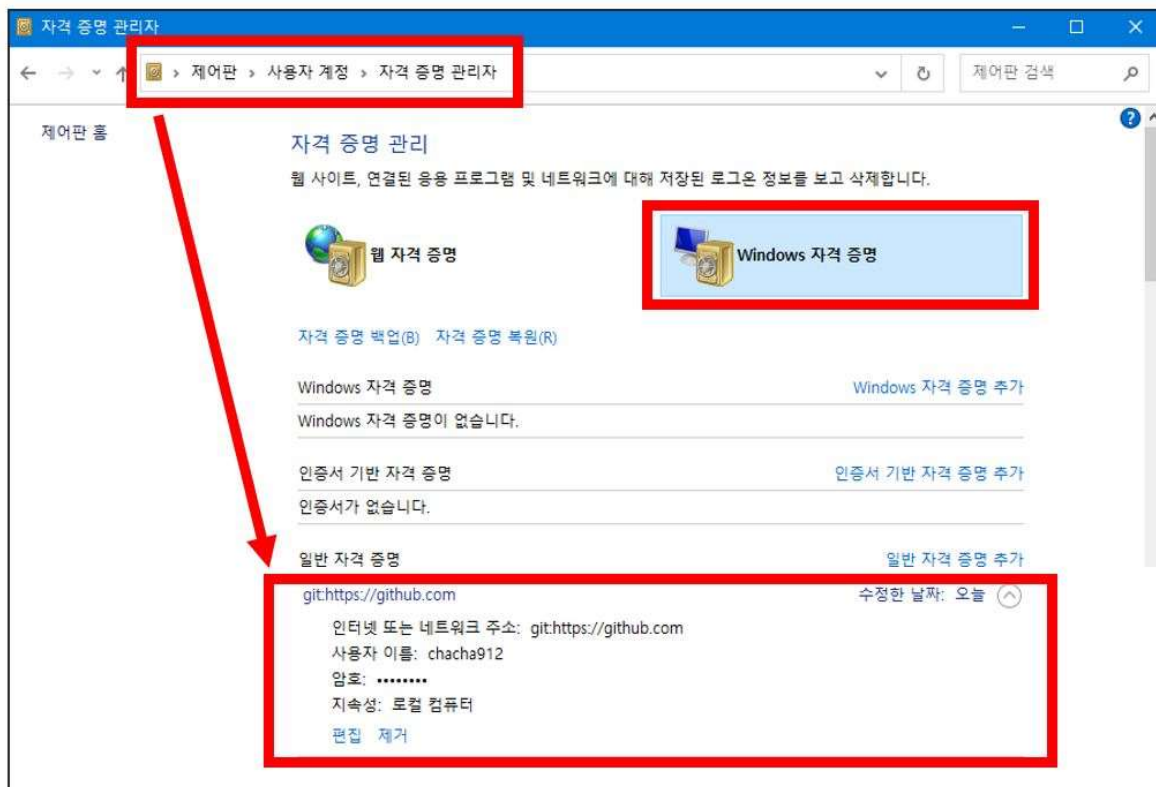
- [제어판] > [사용자 계정] > [자격 증명 관리자] > [Windows 자격 증명]



- 목록에서 [일반 자격 증명]에서 GitHub 항목이 없으면 [일반 자격 증명 추가]를 클릭하여 다음과 같이 각각의 값을 입력하고 [확인] 버튼을 클릭한다.
 - [인터넷 또는 네트워크 주소] : git:https://github.com
 - [사용자 이름] : GitHub 계정 이름
 - [암호] : Access Token 값



- 만일 기존의 값이 있다면 다음 그림을 참고하여 수정한다.



[<https://velog.io/@rimo09/Github-%EC%9C%88%EB%8F%84%EC%9A%B0-git-credential-access-token-%EC%A0%81%EC%9A%A9%ED%95%98%EA%B8%B0>]

18) Command Line 에서 Access Token 을 사용할 때는 다음 그림을 참고한다.

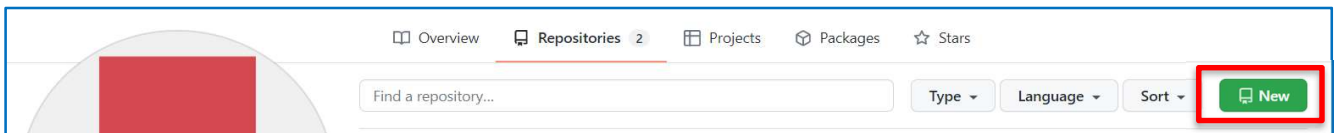
```
$ git clone https://HOSTNAME/USERNAME/REPO.git
Username: YOUR_USERNAME
Password: YOUR_TOKEN
```

19) Personal Access Token 은 HTTPS Git 작업에만 사용할 수 있다.

20) 혹시 Username 과 Password 가 표시되지 않으면 자격 증명이 컴퓨터에서 Cache 되어 있을 수 있다. macOS 나 Windows 의 자격 증명을 업데이트하여 이전 암호를 Token 으로 변경한다.

6. Repository 생성

1) [New]를 click 하여 새 Repository 를 생성한다.



2) [Repository name]에 저장소 이름을 'Hello-World'라고 입력한다.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

gitinstructor

 /

Hello-World

Great repository names are short and memorable. Need inspiration? How about **ideal-giggle**.

Description (optional)

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

Create repository

- 3) **[Description]** : 저장소에 대한 설명을 입력한다.
- 4) 필수는 아니기 때문에 공백으로 놓아도 된다.
- 5) **[Public]**과 **[Private]** : 저장소의 공개 여부이다.
- 6) **[Private]**을 선택하면 비공개 저장소가 생성된다.
- 7) **[Initialize this repository with a README]** : 이것을 check 하면 GitHub 저장소 초기화와 README.md file 이 자동으로 설치된다.
- 8) 이 경우, 저장소 작성 직후 바로 clone 으로 local 저장소에 복제가 가능하다.
- 9) 이미 만들어 놓은 Git repository 를 GitHub 에 등록하고 싶은 경우에는 Check 하지 않고 직접 push 할 것을 권장한다.
- 10) **[Add .gitignore]** : 이 Button 을 사용하면 초기화할 때 .gitingore File 을 작성해 준다.
- 11) .gitignore File 에는 Version 을 관리하지 않아도 되는 File 또는 Directory 가 설정되기 때문에, 사용하는 언어 또는 Framework 이나 Library 에서 별도의 설정을 하지 않아도 되는 장점이 있다.
- 12) Dropdown menu 에는 주요한 언어와 Framework 가 있다.
- 13) 여기에서 사용할 언어 또는 Framework 를 선택하면 된다.
- 14) 한번 'Java'를 선택해 보았다.
- 15) .gitignore File 을 어떻게 만들어야 할 지 잘 모르면 <https://www.gitignore.io> 를 방문해서 확인하기 바란다.
- 16) 사용하는 OS, IDE, Programming 언어를 입력하면 자동으로 .gitignore File 을 생성해 준다.
- 17) **[Add a license]** : License 를 선택할 수 있다.
- 18) 이 저장소에 필요한 License 를 선택하는 것이다.
- 19) License 를 선택하고 저장소를 생성하면 License 의 내용이 적인 LICENSE File 이 저장소 내부에 생성된다.
- 20) 이 File 로 저장소에 있는 Code 들의 License 를 알려주는 것이다.
- 21) 참고로, Code 를 공개할 때의 License 에 대해 알아보자.
- 22) 다음은, <소셜 코딩으로 이끄는 GitHub 실천기술> p.41 에 있는 내용을 발췌했다.

GitHub 에서 소스 코드를 공개한다고 해도 저작권 등을 포기하는 것을 아니며, 라이선스는 코드의 권리 소유자가 적절한 것을 선택해야 합니다. GitHub 에서는 수정 BSD 라이선스 또는 Apache 라이선스 등 다양한 라이선스가 선택되고 있지만, 대부분의 소프트웨어는 MIT 라이선스를 이용하고 있습니다. MIT 라이선스의 특징은 다음과 같습니다.

이 소프트웨어를 누구라도 무상으로 제한 없이 취급해도 좋다. 단, 저작권 표시 및 이 허가 표시를 소프트웨어의 모든 복제물 또는 중요한 부분에 기재해야 한다.

저자 또는 저작권자는 소프트웨어에 관해서 아무런 책임을 지지 않는다.

-[MIT 허가서] "Wikipedia 'http://ko.wikipedia.org/ 2013 년 3 월 10 일 최종 변경

자세한 내용은 원문을 확인해 주세요.

실제로 라이선스를 이용하는 방법은 굉장히 간단합니다. LICENSE 파일 등을 리포지토리에 두고, README.md 파일에서 어떤 라이선스를 사용하고 있는지 표시해 주면 됩니다.

라이선스가 표시되어 있지 않은 소프트웨어를 사용할 때는 만약을 위해 저작자에게 직접 문의할 것을 추천합니다.


23) 입력이 끝났으면 **[Create repository]** 초록색 Button 을 Click 한다.

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 *

 gitinstructor ▾


 /

Hello-World 


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

Description (optional)

Hello-World 저장소 생성

☒  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: Java ▾

☒ Choose a license

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

License: MIT License ▾

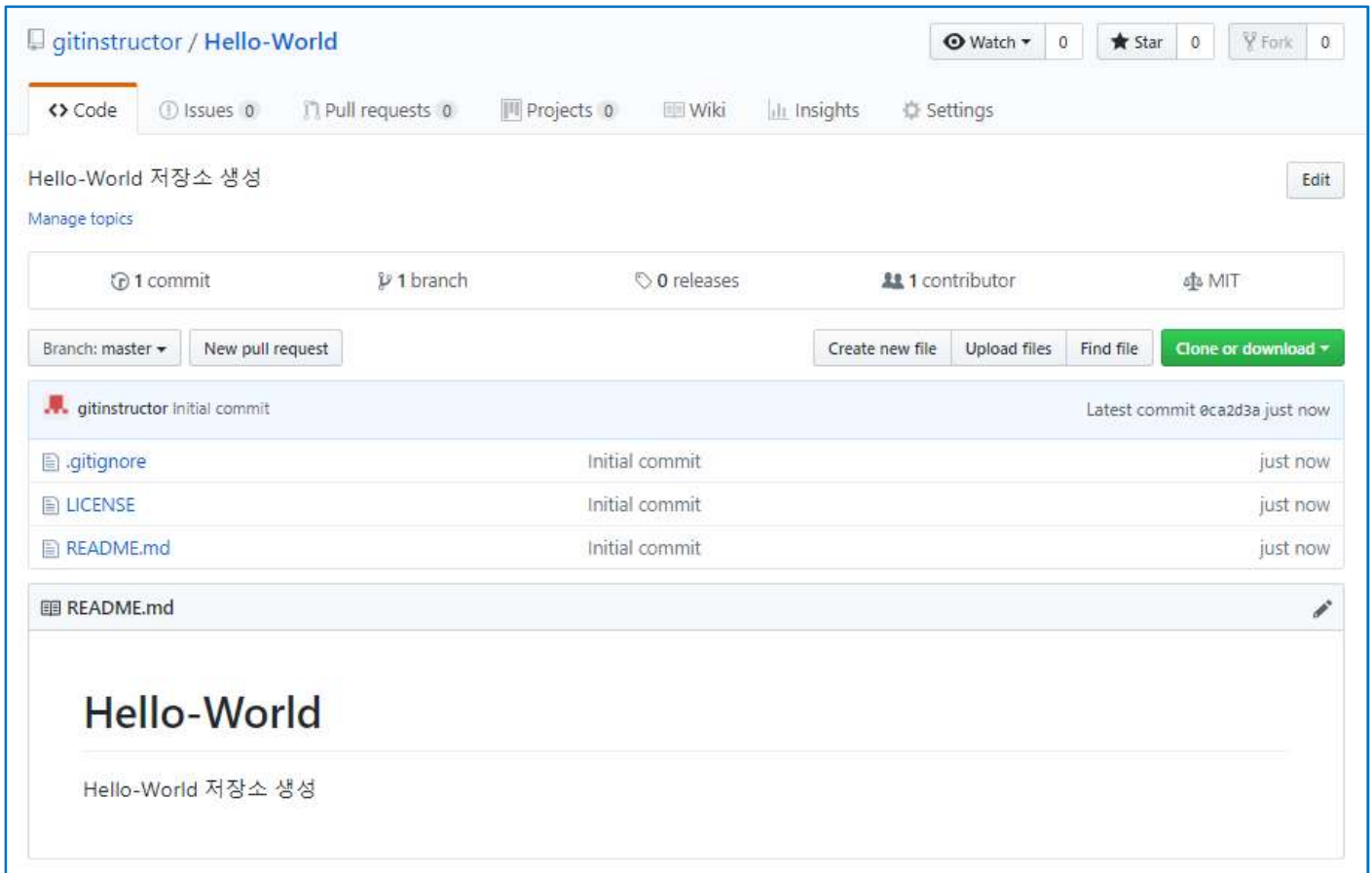
This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

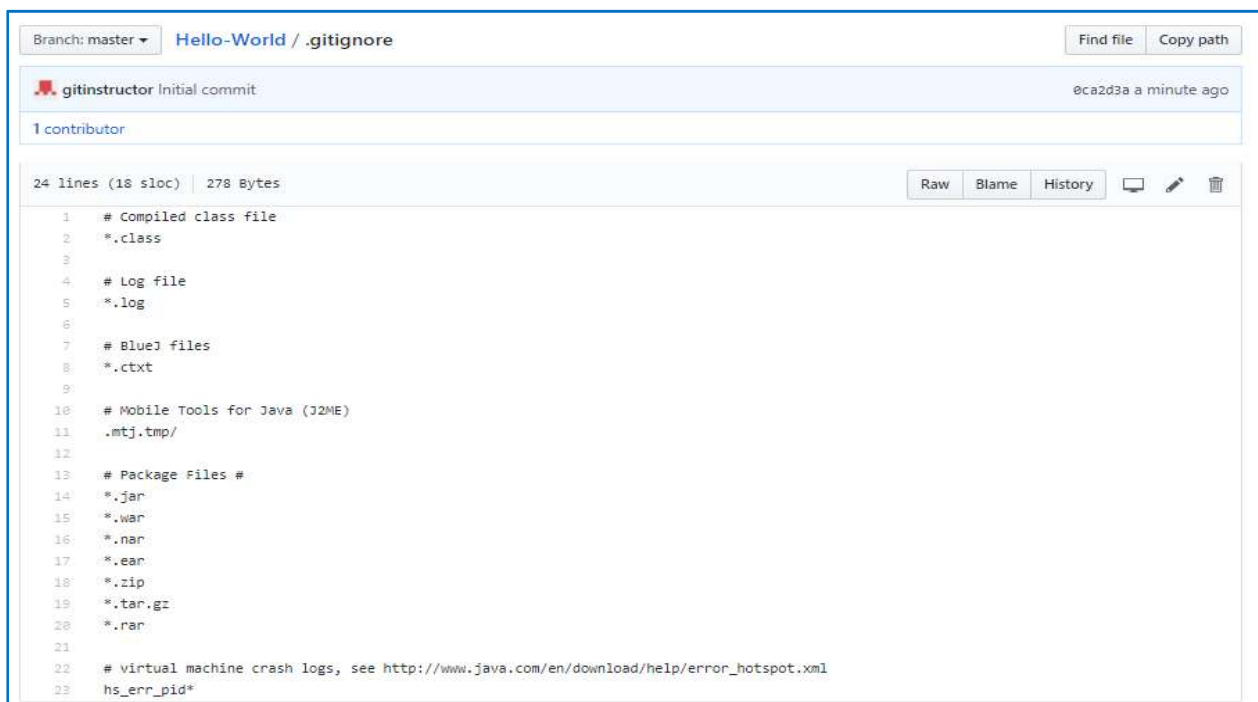
24) 방금 생성한 저장소의 접근 URL 은 다음과 같다.

<https://github.com/사용자이름/Hello-World>

25) 아래의 그림은 방금 생성한 저장소의 모습이다.




26) 다음은 .gitignore file 의 내용이다. Java 를 선택했었다.




27) #은 주석이다.

28) 다음은 License file 의 내용이다. MIT License 를 선택했었다.

Branch: masterHello-World / LICENSEFind fileCopy path

 **MIT License**
gitinstructor/Hello-World is licensed under the MIT License
A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.
[This is not legal advice. Learn more about repository licenses.](#)

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	① License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

 **gitinstructor** Initial commit0ca2d3a 3 minutes ago

1 contributor

22 lines (17 sloc) | 1.04 KBRawBlameHistory

```
1 MIT License
2
3 Copyright (c) 2018 gitinstructor
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
```

29) 마지막으로, README.md File 은 초기화할 때 생성되었다.

30) README.md File 은 저장소의 최상위 Page 에 자동으로 표시된다.

31) 이 File 은 Markdown 문법으로 작성된다.

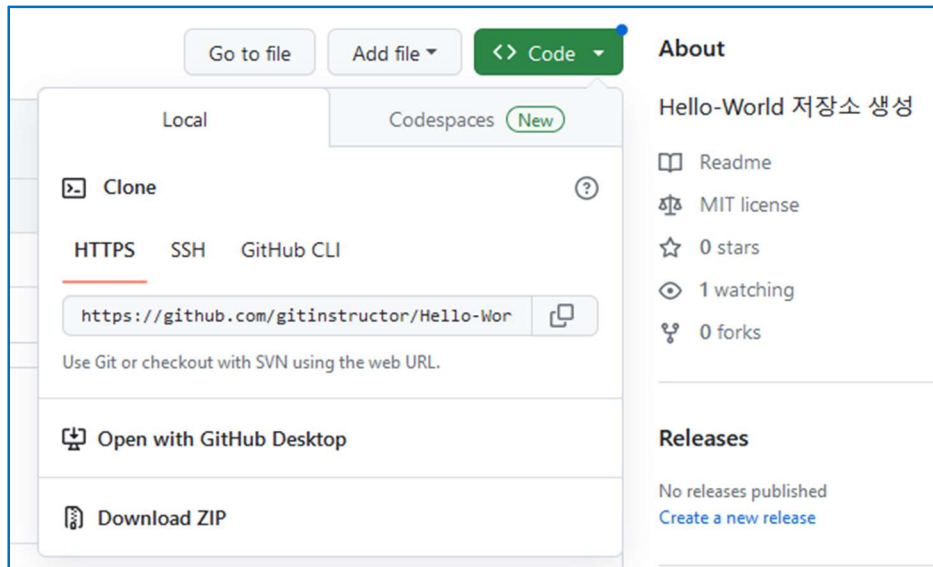
Hello-World / README.md in mainCancel changes

<> Edit filePreviewSpaces2Soft wrap

```
1 # Hello-World
2 Hello-World 저장소 생성
3
```

7. 생성된 Repository Clone 하기

- 1) 생성된 저장소에 실제로 Code 를 작성하고 공개해 보자.
- 2) 일단 작성한 Repository 를 Clone 해서 개발 환경을 구성해야 한다.
- 3) Clone 의 주소는 다음 그림과 같다.



- 4) 위의 주소로 Clone 해보자.

```
GitWork — -bash — 97x24
henry-MacBook-Pro:GitWork henry$ git init
Initialized empty Git repository in /Users/henry/GitWork/.git/
henry-MacBook-Pro:GitWork henry$ git config user.name gitinstructor
henry-MacBook-Pro:GitWork henry$ git config user.email devtrainer@naver.com
henry-MacBook-Pro:GitWork henry$ git clone https://github.com/gitinstructor/Hello-World.git
Cloning into 'Hello-World'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
henry-MacBook-Pro:GitWork henry$ ls -al
total 0
drwxr-xr-x  4 henry  staff  128 Apr  5 11:41 .
drwxr-xr-x+ 55 henry  staff 1760 Apr  5 09:36 ..
drwxr-xr-x  9 henry  staff  288 Apr  5 11:40 .git
drwxr-xr-x  6 henry  staff  192 Apr  5 11:41 Hello-World
henry-MacBook-Pro:GitWork henry$
```

- 5) 이 저장소에서 공개되는 Code 를 Commit 해서 GitHub 의 저장소에 Push 하면 Code 가 공개된다.

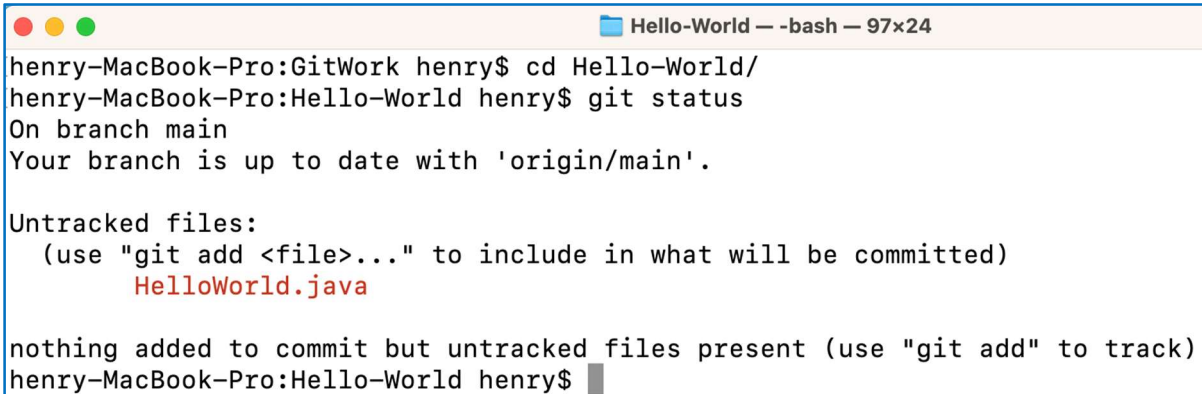
8. Code 작성

1) 일단은, 간단하게 test 하기 위해 HelloWorld.java 를 작성하기로 하자.

```
package com.example;
```

```
public class HelloWorld {  
    public static void main(String [] args){  
        System.out.println("Hello, World");  
    }  
}
```

2) 현재 Status 는 아래와 같다.

A terminal window titled "Hello-World — -bash — 97x24" showing the output of the 'git status' command. The output indicates the current branch is 'main' and it is up to date with 'origin/main'. It also lists 'Untracked files:' as 'HelloWorld.java' and provides instructions to use 'git add' to track the file. The prompt shows the user is in the 'Hello-World' directory.

```
henry-MacBook-Pro:GitWork henry$ cd Hello-World/  
henry-MacBook-Pro:Hello-World henry$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    HelloWorld.java  
  
nothing added to commit but untracked files present (use "git add" to track)  
henry-MacBook-Pro:Hello-World henry$
```

9. Commit 하기

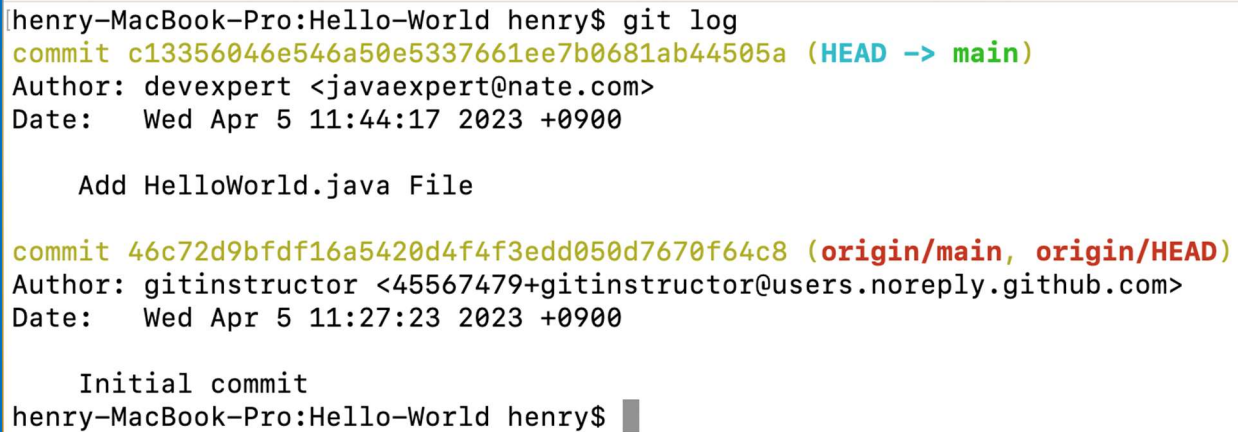
1) 위에서 생성한 HelloWorld.java 를 Commit 해 보자.

2) 먼저 Staging Area 에 Add 한다.

```
$ git add HelloWorld.java
```

3) 그리고 Commit 하고 Log 를 확인해 보자.

```
$ git commit -m "Add HelloWorld.java File"
```

A terminal window titled "Hello-World - -bash - 97x24" showing the output of the 'git log' command. The output displays two commits. The first commit, with hash c13356046e546a50e5337661ee7b0681ab44505a, is labeled '(HEAD -> main)' and was authored by devexpert on Wed Apr 5 11:44:17 2023. The second commit, with hash 46c72d9bdf16a5420d4f4f3edd050d7670f64c8, is labeled '(origin/main, origin/HEAD)' and was authored by gitinstructor on Wed Apr 5 11:27:23 2023. The commit message for the second commit is "Initial commit". The terminal prompt is currently at the second commit's details.

```
henry-MacBook-Pro:Hello-World henry$ git log
commit c13356046e546a50e5337661ee7b0681ab44505a (HEAD -> main)
Author: devexpert <javaexpert@nate.com>
Date:   Wed Apr 5 11:44:17 2023 +0900

    Add HelloWorld.java File

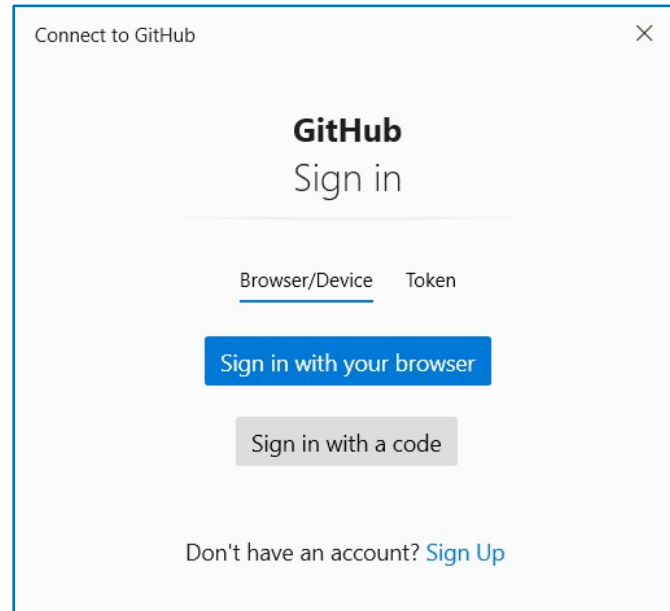
commit 46c72d9bdf16a5420d4f4f3edd050d7670f64c8 (origin/main, origin/HEAD)
Author: gitinstructor <45567479+gitinstructor@users.noreply.github.com>
Date:   Wed Apr 5 11:27:23 2023 +0900

    Initial commit
henry-MacBook-Pro:Hello-World henry$ █
```

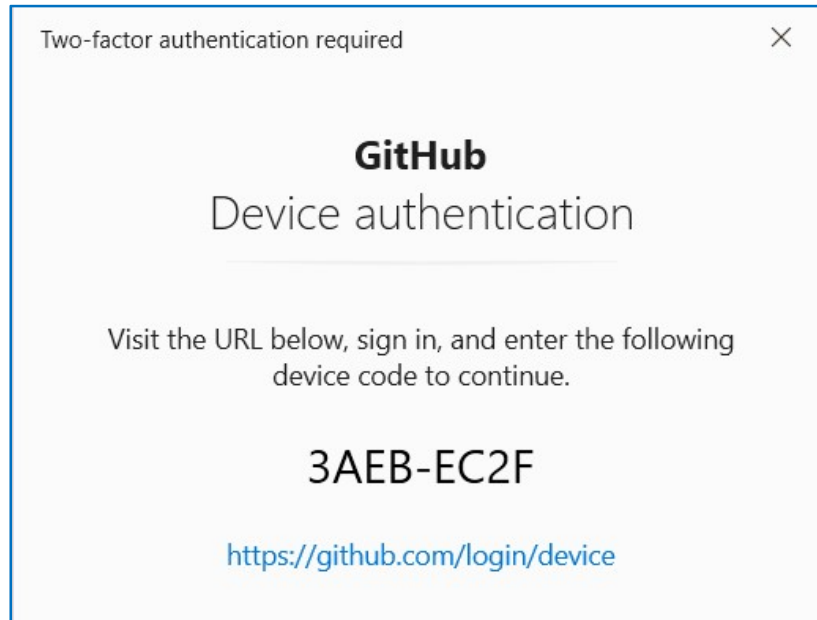
10. push 하기

- 1) Windows 에서 다음과 같이 Push 하면 먼저 Login 을 하라고 하고, Login 인증이 끝나면 GitHub 에 있는 저장소가 갱신된다.

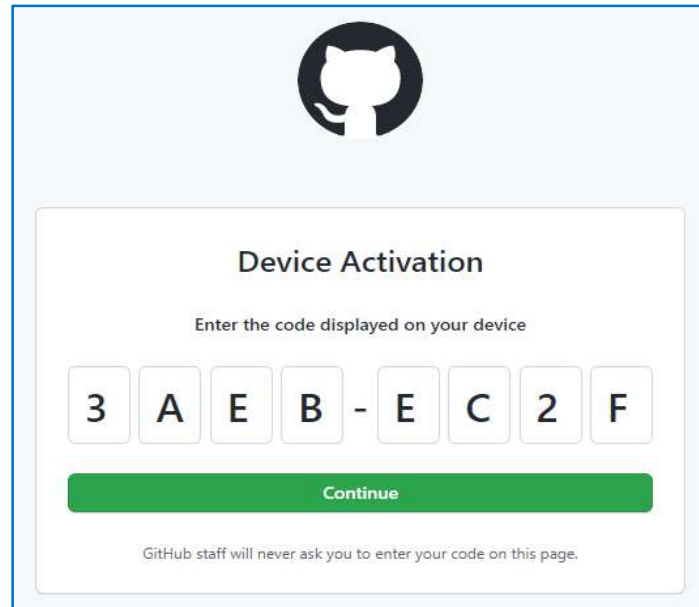
\$ git push



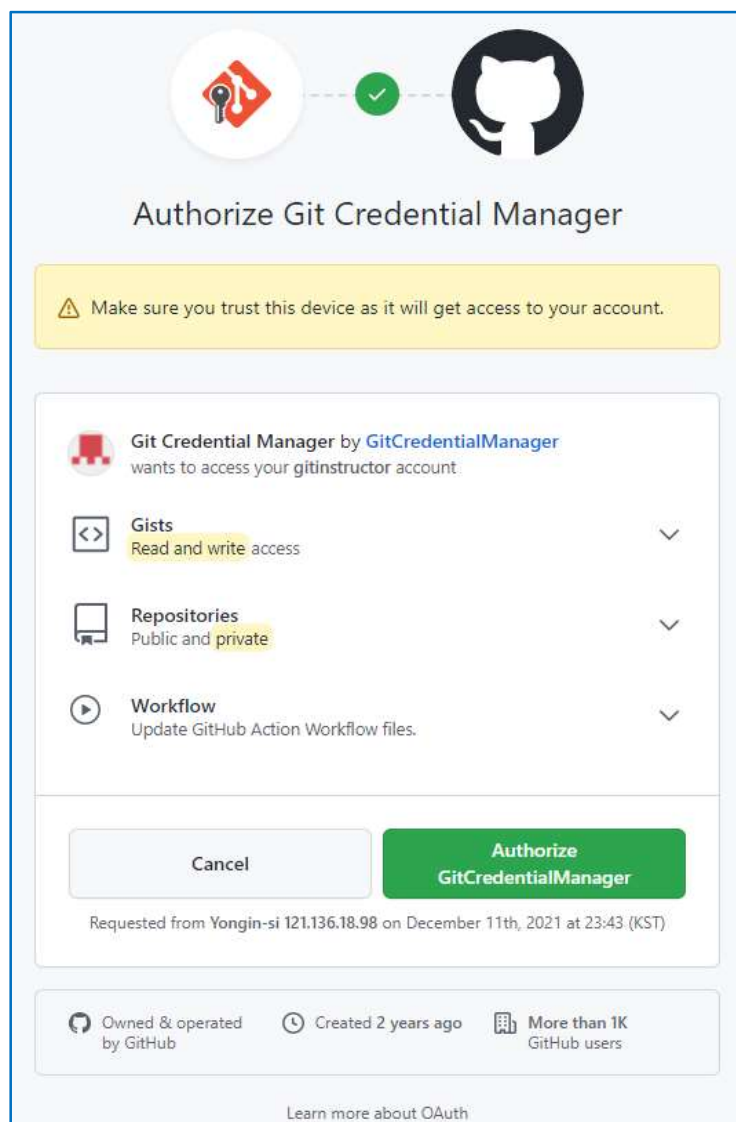
- 2) 인증을 위해 **[Sign in with a code]**를 클릭한다.



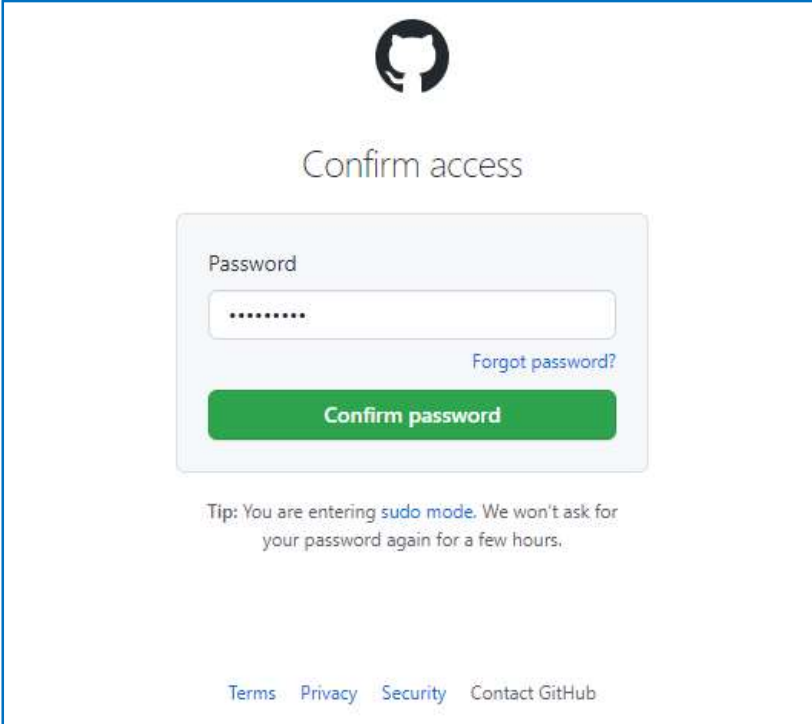
- 3) GitHub Device authentication 창에서 아래 링크를 방문하여 인증코드를 넣고, **[Continue]** 초록색 버튼을 클릭한다.



- 4) **[Authorize GitCredentialManager]** 초록색 버튼을 클릭한다.

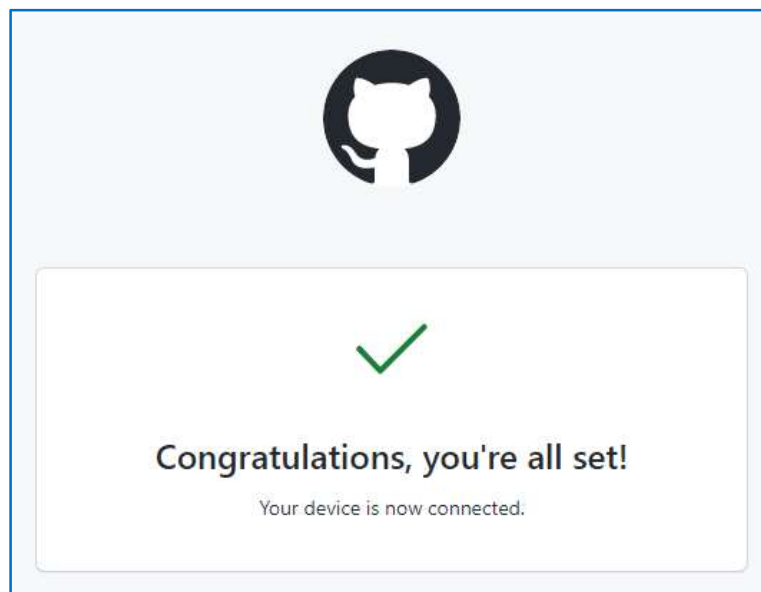


5) 인증을 위해 패스워드를 입력하고 **[Confirm password]** 버튼을 클릭한다.



The image shows the GitHub 'Confirm access' screen. At the top is the GitHub logo. Below it, the text 'Confirm access' is centered. A light blue box contains a 'Password' label, a password input field with masked characters, a 'Forgot password?' link, and a green 'Confirm password' button. Below the box, a tip states: 'Tip: You are entering **sudo mode**. We won't ask for your password again for a few hours.' At the bottom are links for 'Terms', 'Privacy', 'Security', and 'Contact GitHub'.

6) 인증작업이 성공적으로 끝났다.



7) macOS에서는 이미 KeyChain에 추가했기 때문에 별다른 작업을 하지 않아도 된다. Push 작업도 성공적으로 끝났다.

- Username :
- Password : Github에서 생성한 Personal Access Token 값

```
henry-MacBook-Pro:Hello-World henry$ git push
Username for 'https://github.com': gitinstructor
Password for 'https://gitinstructor@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 475 bytes | 475.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/gitinstructor/Hello-World.git
    46c72d9..c133560  main -> main
henry-MacBook-Pro:Hello-World henry$
```

8) 실제로 해당 저장소에 들어가서 확인해 보자. 성공적으로 HelloWorld.java 파일이 Upload 되었다.

