

1. Git 그리고 GitHub

1.1. 버전 관리란?

예를 들어 팀 프로젝트에서 사용할 공용 폴더를 만들고 그곳에 소스 코드를 올려 놓는다고 가정해 봅시다. 이 곳에 올려진 최초의 소스코드를 '00 버전'이라고 합시다. 팀원 A가 '00 버전'의 소스코드를 수정해서 '01 버전'으로 저장하고, 팀원 B가 '01 버전'을 수정해서 '02 버전'으로 저장할 수 있을 겁니다. 이렇게 최초 버전을 기준으로 수정하고 순차적으로 번호를 붙여서 저장하며 버전 관리가 됩니다. 그런데 팀원 B와 팀원 C가 동시에 '01 버전'을 수정해서 '02 버전'으로 저장한다면 어떻게 될까요?

팀원 프로젝트에 참여하는 인원이 많을수록, 프로젝트 기간이 길수록 어느 파일이 최종 업데이트 파일인지 확인할 길이 막막해집니다. 그래서 여럿이 함께 작업하는 협업 프로젝트에는 버전 관리가 필요합니다. 이것을 바로 버전 관리라고 합니다.

1.2. Git 그리고 Github

내가 원하는 시점마다 깃발을 꽂고, 깃발이 꽂힌 시점으로 자유롭게 이동할 수 있다면 편안하게 새로운 소스코드를 추가하거나 삭제할 수 있을 겁니다. 소스코드 오류가 일어난다면 바로 전제 꽂은 깃발 시점으로 돌아가면 되니까요. 이를 가능하게 해 주는 소스코드 버전 관리 시스템이 바로 **Git**입니다. Git은 소스코드 버전 사이를 오가는 시간 여행 이상의 기능을 제공합니다.

Git은 데이터를 저장할 공간만 있다면 어디서나 사용할 수 있습니다. 개인 컴퓨터에만 저장한다면 나 혼자 사용할 수 있겠죠? USB에 저장한다면 휴대하면서 어디서든 내가 작업하던 프로젝트를 사용할 수 있을 겁니다. 만약 드롭박스, 구글 드라이브와 같은 클라우드 서버에 올려둔다면, 어떨까요? 팀 프로젝트를 하는 다른 팀원과 함께 인터넷을 통해 버전 관리를 할 수 있을 겁니다.

이렇게 Git으로 관리하는 프로젝트를 올려둘 수 있는 Git 호스팅 사이트 중 하나가 바로, Github입니다. 블로그를 만들 수 있는 곳이 네이버, 다음, 워드프레스 등 다양한 것처럼 Git으로 관리하는 프로젝트를 올릴 수 있는 사이트도 Github뿐만 아니라 GitLab, BitButcket등 다양합니다.

Git 호스팅 사이트	모기업	특징	가격 정책
Github.com	GitHub inc. (Microsoft에서 인수)	사용자 2,800만 명. 세계 최대 규모의 Git 호스팅 사이트	공개저장소 생성 무료. 비공개저장소는 작업자 3인 이하인 경우에는 무료, 설치형 버전인 Enterprise를 월 21달러에 사용할 수 있다.
GitLab.com	GitLab.inc	GitHub에 뒤지지 않는다. NASA,	공개저장소 및 비공개

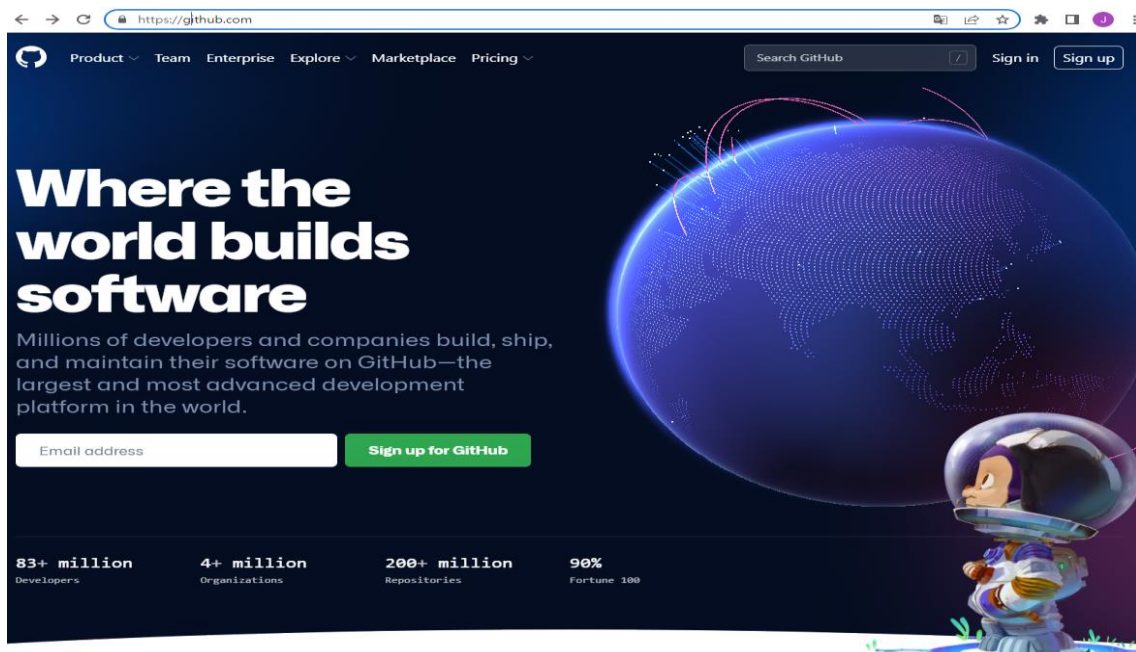
		Sony 등 10만 개 이상의 조직이 사용하고 있다. GitLab 프로젝트 자체가 오픈소스여서 직접 서비스 발전에 기여할 수 있다.	저장소 생성 무료. 소스 코드 빌드에 유용한 도구 자원 성능에 따라 월 4달러 ~ 99달러 부담
BitBucket.org	Atlassian	사용자 600만 명. 이슈관리 시스템인 지라(Jira)를 만든 Atlassian 이 모기업 이어서 지라와 연동이 쉽다.	5명 이하 팀이면 공개 저장소 및 비공개저장소 생성 무료. 그 이상이면 월 2달러 ~ 5달러 부담

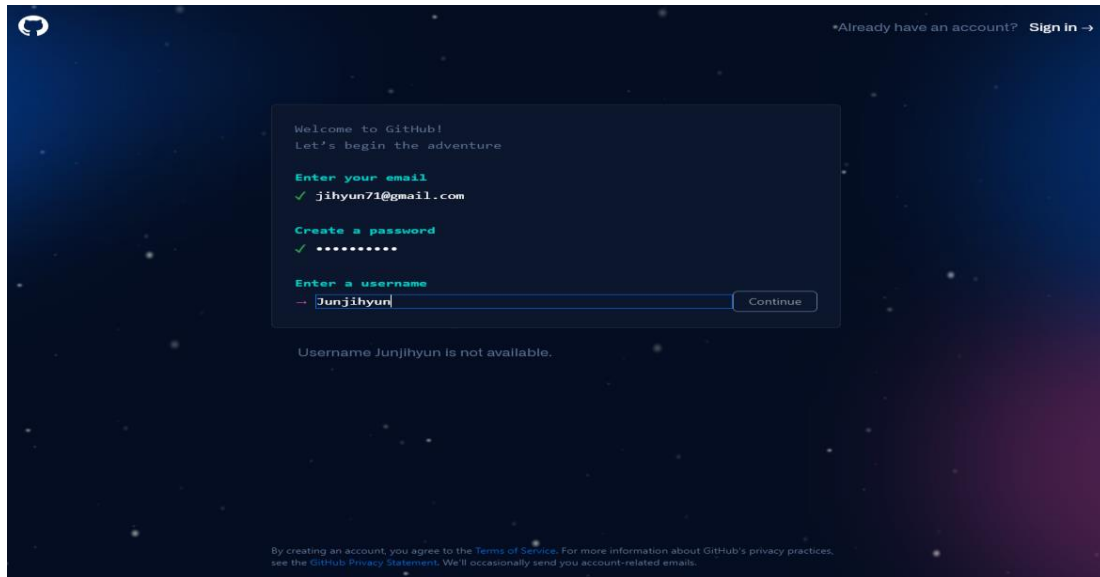
GitHub에 소스코드를 올려두면 시간, 공간의 제약 없이 협업할 수 있다. 또한 프로젝트를 공개저장소로 만들면 이름도, 얼굴도 모르는 전 세계 개발자와 협업할 수 있다. 이렇게 누구든지 기여할 수 있는 공개저장소 프로젝트를 오픈소스라고 한다.

1.3. GitHub 가입하기

GitHub 사이트(<http://GitHub.com/>)에 접속해서 [Sing up]버튼을 클릭한 후, UserName, Email, Password를 입력하고 초록색 [Sing up for GitHub] 버튼을 클릭한다.

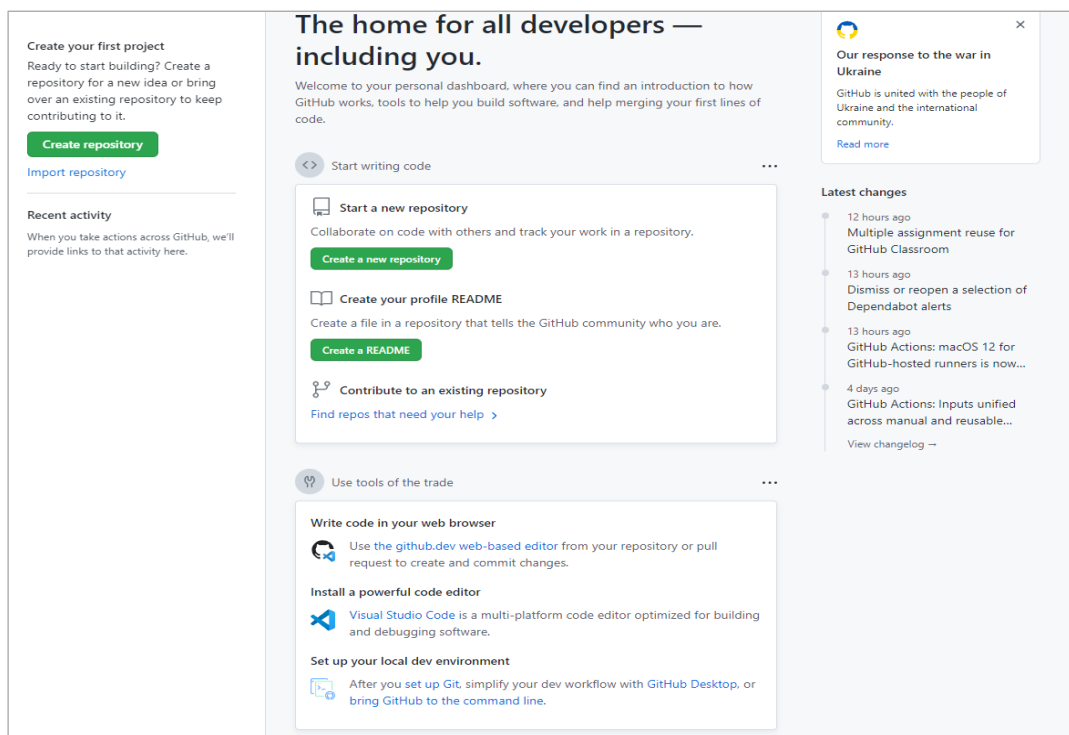
Email 기재 시 실제 사용이 가능한 Email 주소를 기입해야 한다. 이유는 GitHub에서 확인 메일을 보내기 때문이다.





위의 과정을 진행하다보면 GitHub를 무료 버전으로 사용할지 아니면 매월 비용을 지불하는 유료 버전으로 사용할지 선택하는 페이지가 나오게 된다. 기본값인 [free]를 선택하고, 다른 옵션은 그대로 두고 가장 아래에 있는 [Continue] 버튼을 클릭한다.

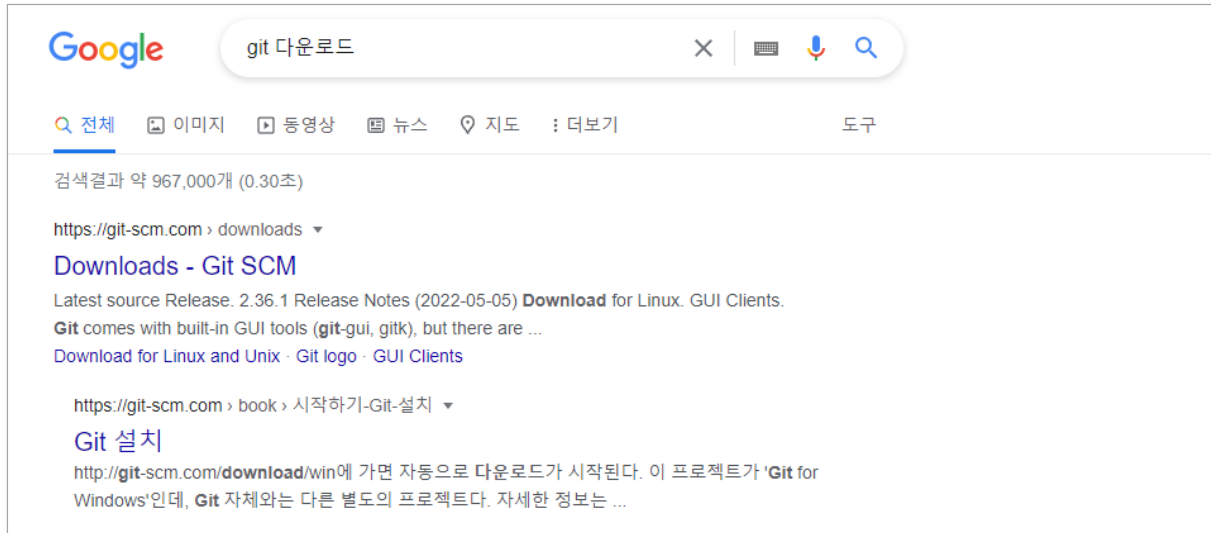
그러면 GitHub에서 메일 주소 검증 위한 메일을 발송하게 된다. 가입이 완료되고 나면 아래와 같은 화면이 뜨게 된다.



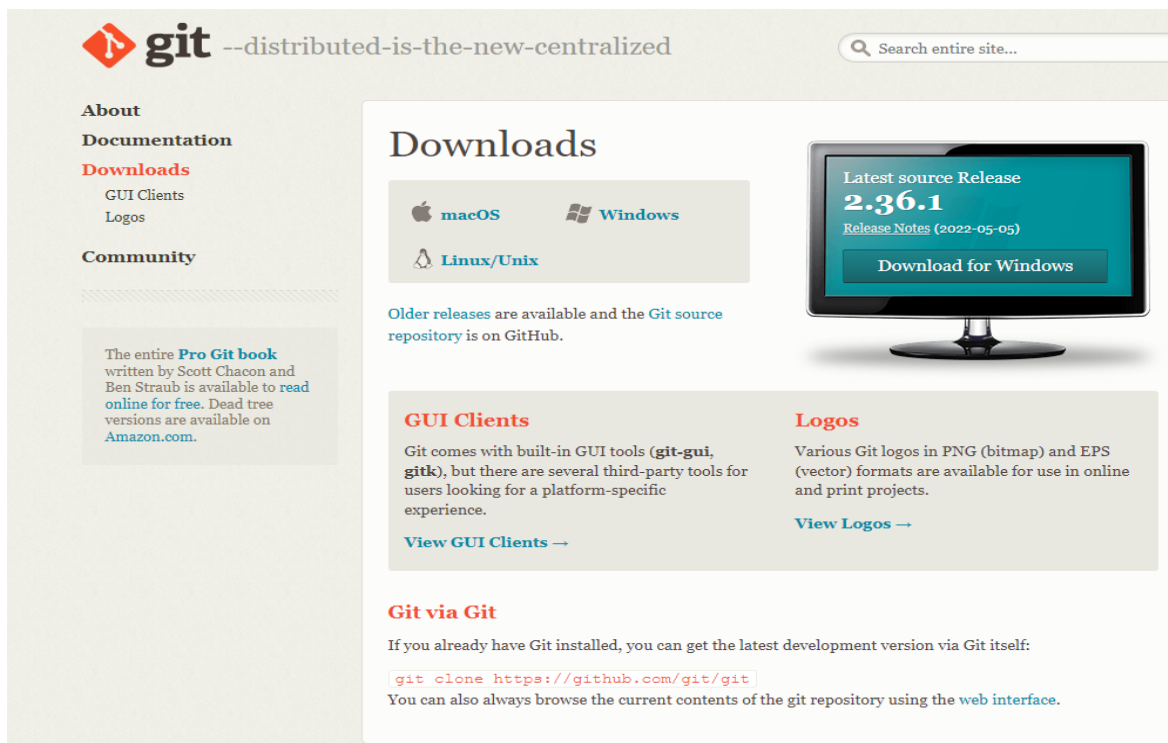
2. Git을 설치하고 로컬저장소에서 커밋 관리하기

2.1. 내 컴퓨터에 Git 설치하기

구글 검색창을 열고 "Git 다운로드"라고 검색하고 'https://git-scm.com/downloads' 링크로 접속한다.



Git 다운로드 페이지에 접속해서 Downloads 항목에서 본인의 운영체제에 맞는 링크를 클릭한다. 만약 윈도우 운영체제이면 [Download 2.36.1 for Windows]를 클릭한다.



Download for Windows

[Click here to download](#) the latest (2.36.1) 64-bit version of **Git for Windows**. This is the most recent maintained build. It was released **about 1 month ago**, on 2022-05-09.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.36.1**. If you want the newer version, you can build it from [the source code](#).

Now What?

Now that you have downloaded Git, it's time to start using it.



Read the Book

Dive into the Pro Git book and learn at your own pace.



Download a GUI

Several free and commercial GUI tools are available for the Windows platform.

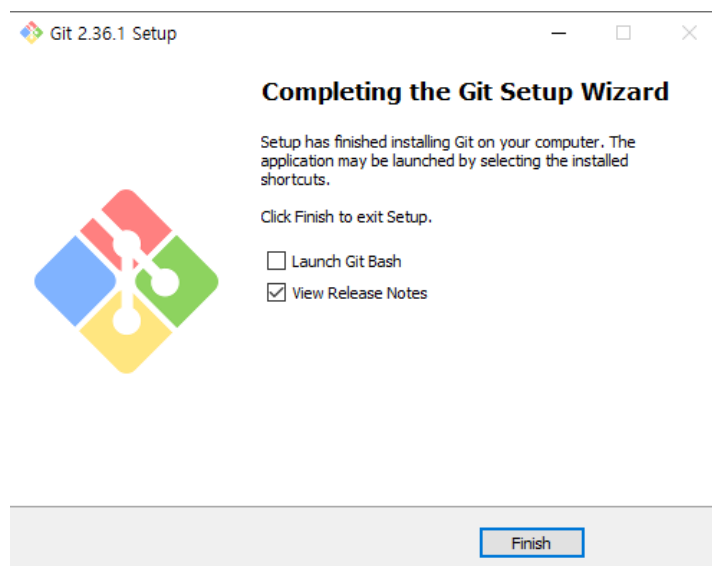


Get Involved

A knowledgeable Git community is available to answer your questions.

위의 그림처럼 본인의 OS 운영체제에 맞는 것을 다운받아 설치를 한다.

다운받은 Git 설치 파일을 실행하면 아래와 같이 설치 창이 계속 뜨게 되며, 기본 설정을 유지한 채 [Next] 버튼 및 [Install] 버튼을 클릭해서 마지막 설치 화면까지 진행하면 된다.

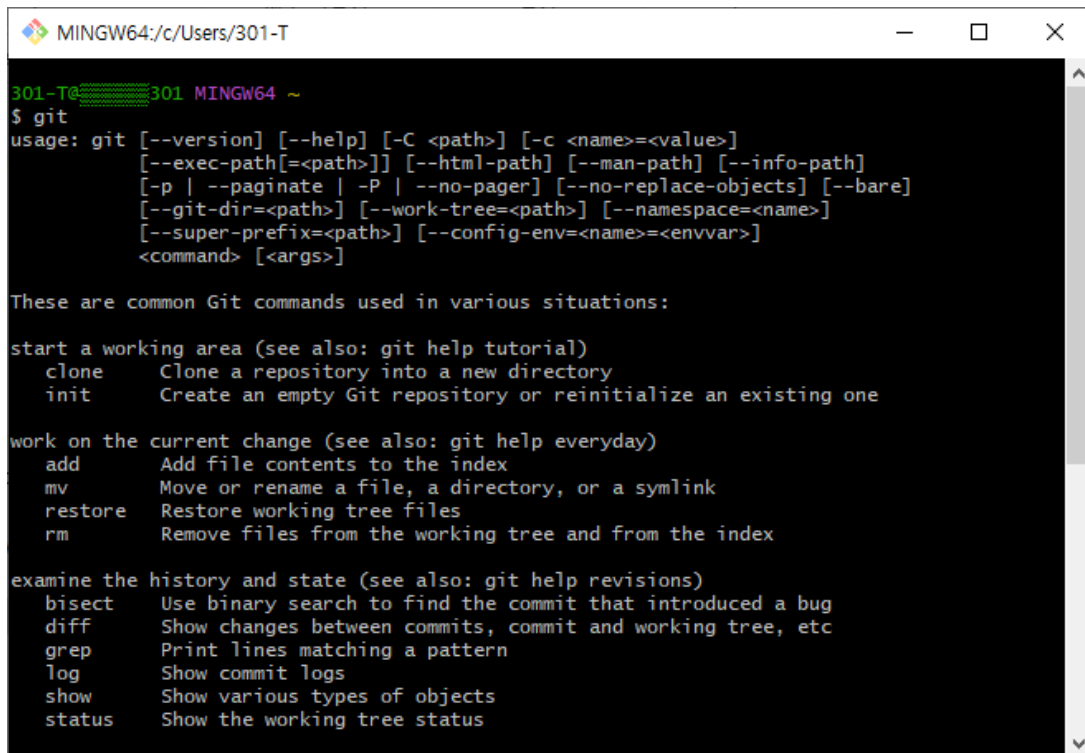


설치 완료 후 위의 처럼 설치 완료 화면이 나오면 [view Release Notices]를 체크하지 않고 [Finish] 버튼을 클릭해서 설치를 완료한다.

Git이 잘 설치되었는지 확인하기 위해 Git Bash를 실행해 보자. 윈도우 시작 버튼 옆에 있는 돋보기 아이콘을 클릭해서 'Git Bash'를 입력하고, 찾아서 실행해본다.

그러면 검정화면이 뜰 것이다. '\$' 표시 옆에 'git'이라고 입력한 후 'enter'키를 친다.

아래 화면처럼 Git 기본 명령어에 대한 안내가 나오면 Git이 제대로 설치된 것이다.



```
301-T@301 MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status
```

2.2. 로컬저장소 만들기

내 컴퓨터에 설치한 Git과 연결할 로컬저장소를 만들어 보겠습니다. 로컬 저장소는 실제로 Git을 통해 버전 관리가 이뤄질 내 컴퓨터에 있는 폴더입니다.

1) 'D:/Programming/cat-program' 이름의 폴더를 내 컴퓨터에 만듭니다. 이 폴더에 프로그램을 할 예정입니다.

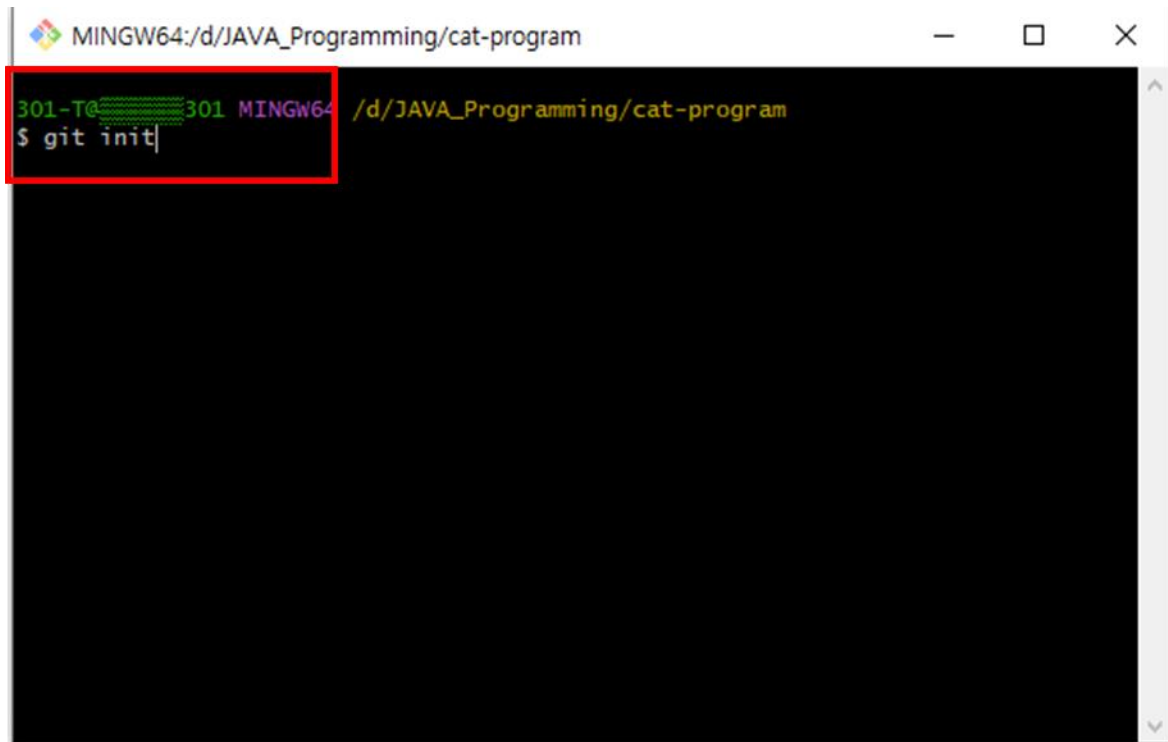
2) 만들어진 폴더안에 'README.txt'라는 파일을 만듭니다.

3) 만들어진 텍스트 파일을 열고 '개발자 티셔츠 쇼핑물 오픈소스'라고 적어주고 저장을 합니다.

4) [cat-program] 폴더에서 마우스 오른쪽 버튼을 클릭하고 [Git Bash Here]를 클릭합니다.

5) Git Bash 창이 열리는데, '\$' 기호는 명령어 입력을 기다리는 커서입니다. 이 기호 옆에 다음과

같은 명령어를 입력합니다.



```
MINGW64:/d/JAVA_Programming/cat-program
301-T@301 MINGW64 /d/JAVA_Programming/cat-program
$ git init
```

'Initialized empty Git repository'라는 텍스트가 나오면 성공입니다. 즉, 처음 만든 것이기 때문에 로컬 저장소는 만들어지나 그 안이 비어있기 때문에 'empty'라는 결과 나오는 것이 맞습니다.

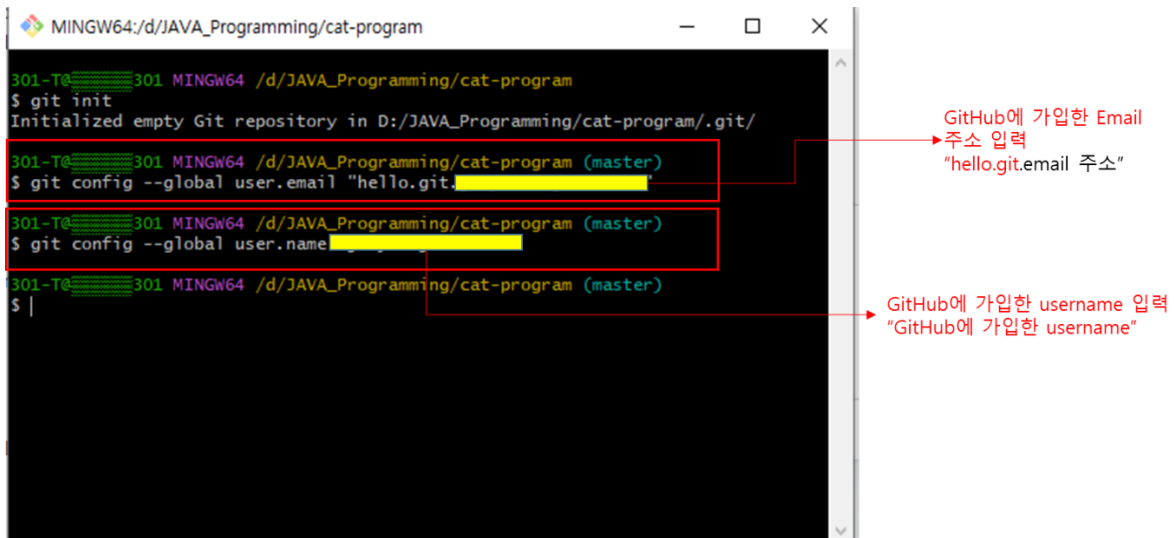
이 명령어를 실행하게 되면 [cat-program] 폴더 안에 [.git]라는 폴더가 자동으로 생성되어집니다. [.git] 폴더에는 Git으로 생성한 버전들의 정보와 원격저장소 주소 등이 들어 있게 되고, [.git] 폴더가 바로 로컬 저장소가 됩니다.

2.3. 첫 번째 커밋 만들기

방금 생성한 README.txt 파일을 하나의 버전으로 만들어보겠습니다. Git에서는 이렇게 생성되어진 각 버전을 커밋이라고 부릅니다.

1) 버전 관리를 위해 내 정보를 등록해야 합니다. 각 버전을 누가 만들었는지 알아야 협업하기 편하겠죠?

[cat-program] 폴더에서 마우스 오른쪽 버튼을 클릭하고 [Git Bash Here]를 클릭해서 Bash창을 엽니다. 그리고, 다음과 같이 두 개의 명령어를 입력하여 실행을 합니다. 로컬에서만 버전 관리를 할거라면 GitHub에 등록하지 않은 Email, UserName을 입력해도 되지만 GitHub에서 버전 관리를 해야 하기 때문에 앞에서 설정한 GitHub의 계정 정보를 동일하게 입력해야 합니다.



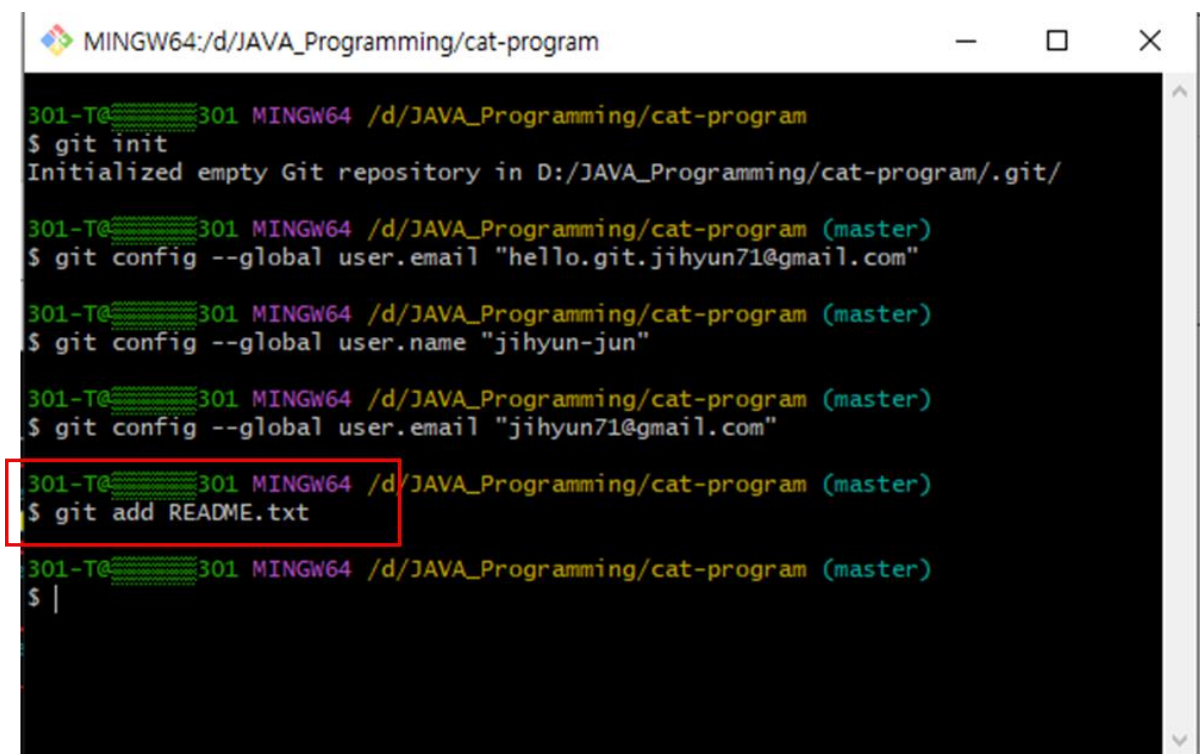
```
MINGW64:/d/JAVA_Programming/cat-program
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program
$ git init
Initialized empty Git repository in D:/JAVA_Programming/cat-program/.git/
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "hello.git. "
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.name " "
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ |
```

GitHub에 가입한 Email 주소 입력
"hello.git.email 주소"

GitHub에 가입한 username 입력
"GitHub에 가입한 username"

[git config - - global user.email "github에 가입한 email 주소"], [git config - -global user.name "github에 가입한 username입력"] 이 두 개의 명령어를 입력하여 실행한다.

그 다음으로 커밋에 추가할 파일을 선택합니다. 조금 전 만들었던 README.txt 파일로 해보겠습니다.



```
MINGW64:/d/JAVA_Programming/cat-program
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program
$ git init
Initialized empty Git repository in D:/JAVA_Programming/cat-program/.git/
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "hello.git.jihyun71@gmail.com"
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.name "jihyun-jun"
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "jihyun71@gmail.com"
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git add README.txt
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ |
```

커밋에는 상세 설명을 적을 수 있습니다. 설명을 잘 적어놓으면 내가 이 파일을 왜 만들었는지, 왜 수정했는지 알 수 있고, 해당 버전을 찾아 그 버전으로 코드를 바꿔 시간 여행을 하기도 수월합니다. '사이트 설명 추가'라는 설명을 붙여서 첫 번째 커밋을 만들어 보겠습니다.


```
MINGW64:/d/JAVA_Programming/cat-program
301-Te 301 MINGW64 /d/JAVA_Programming/cat-program
$ git init
Initialized empty Git repository in D:/JAVA_Programming/cat-program/.git/

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "hello.git.jihyun71@gmail.com"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.name "jihyun-jun"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "jihyun71@gmail.com"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git add README.txt

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git commit -m "사이트 설명 추가"
[master (root-commit) 3bc9a89] 사이트 설명 추가
1 file changed, 1 insertion(+)
create mode 100644 README.txt

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ |
```

[-m]은 'message'의 약자입니다. '1 file changed, 1 insertion(+)' 텍스트가 보이면 성공입니다.

자.. 일단 첫 번째 버전을 만들었습니다. 여기까지 작업하여 성공했다면 README.txt 파일을 수정하고 두 번째 커밋을 해 볼까요? 파일을 열어 맨 뒤에 '짱'이라고 적고 메모장을 저장한 후 닫습니다.

Add 명령어로 README.txt를 선택하고, '설명 업데이트'라는 설명을 붙여서 commit 명령어로 커밋을 만듭니다.

```
MINGW64:/d/JAVA_Programming/cat-program
301-Te 301 MINGW64 /d/JAVA_Programming/cat-program
$ git init
Initialized empty Git repository in D:/JAVA_Programming/cat-program/.git/

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "hello.git.jihyun71@gmail.com"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.name "jihyun-jun"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git config --global user.email "jihyun71@gmail.com"

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git add README.txt

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git commit -m "사이트 설명 추가"
[master (root-commit) 3bc9a89] 사이트 설명 추가
1 file changed, 1 insertion(+)
create mode 100644 README.txt

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git add README.txt

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git commit -m "설명 업데이트"
[master 27a3e50] 설명 업데이트
1 file changed, 1 insertion(+), 1 deletion(-)

301-Te 301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ |
```

'1 file changed. 1 insertion(+), 1 deletion(-)'라는 메시지가 보이면 성공입니다. 이렇게 두 번째 버전을 만든 것입니다.

2.4. 다른 커밋으로 시간 여행하기

개발을 하다가 요구사항이 바뀌어서 이전 커밋부터 다시 개발하고 싶다면 Git을 사용해 그 커밋으로 돌아가면 되겠죠?

현재 README.txt 파일의 내용은 두 번째 만들었던 '설명 업데이트' 커밋이고 텍스트 문서의 내용은 '개발자 티셔츠 쇼핑물 오픈소스 짱'입니다.

이것을 첫 번째 만들었던 커밋 버전인 '사이트 설명 추가'로 돌려 보겠습니다. 첫 번째 버전으로 돌아가면 텍스트 파일의 내용은 '짱' 단어가 없는 '개발자 티셔츠 쇼핑물 오픈소스'가 되겠죠? 먼저 log 명령어로 지금까지 만든 커밋을 확인합니다.

"\$ git log"라고 입력을 한 후 "enter"를 치게 되면 두 번째 커밋, 첫 번째 커밋한 내용을 확인할 수 있습니다.

우리가 되돌리려는 것은 첫 번째 커밋이니 앞 7자리 커밋 아이디를 복사하고 checkout 명령어로 해당 커밋으로 코드를 되돌립니다.

만약 첫 번째 커밋 내용이 "`commit 3bc9a89be3950b5f5a92c2ece45299bd634b3c98`" 이라면 commit 다음에 나온 코드가 아이디이고 이 아이디를 복사하여 "\$ git checkout `3bc9a89be3950b5f5a92c2ece45299bd634b3c98`" 이라고 입력한 후 'enter'를 치면 된다.

실행 후 마지막에 "HEAD is now at 3bc9a89 사이트 설명 추가"라는 텍스트가 보이면 첫 번째 커밋으로 돌아간 것입니다. README.txt 파일을 열어 확인을 합니다.

다시 체크아웃을 해서 최신 커밋인 두 번째 커밋으로 돌아가겠습니다. 첫 번째 했던 것처럼 두 번째 커밋 아이디인 'git checkout 아이디'를 입력해도 되겠지만 '-'를 적어도 됩니다. 이것은 최신 버전으로의 커밋을 의미합니다.

'\$ git checkout -'라고 입력하고 실행한 다음 README.txt 파일을 열어봅니다.

다시 '짱'이라는 글자가 추가된 최신 버전으로 파일이 변경된 것을 확인할 수 있을 것입니다.

3. GitHub 원격저장소에 커밋 올리기

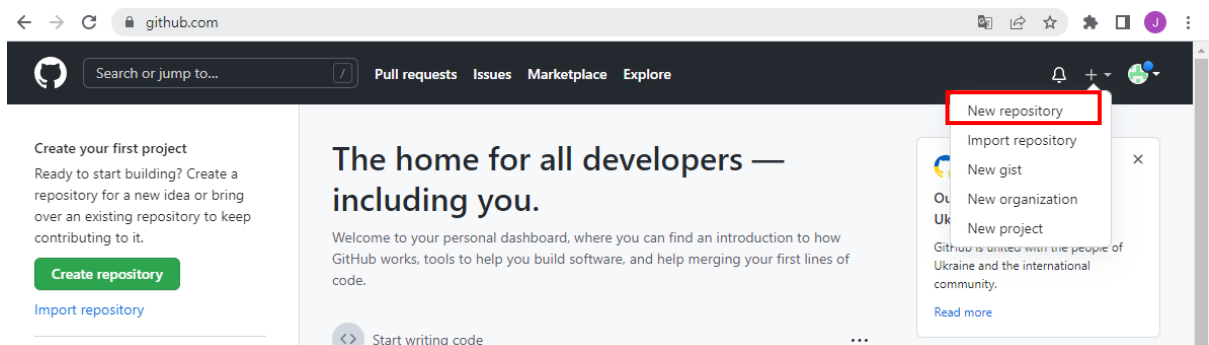
3.1. 원격 저장소 만들기

1) 이번에는 GitHub에 협업할 공간인 원격저장소를 만듭니다. 쉽게 설명하면 GitHub 웹 사이트에

프로젝트를 위한 공용 폴더를 만든 것입니다. 로컬 저장소와 구분하는 개념으로 원격저장소라고 합니다. GitHub에서는 원격저장소를 레포지토리(Repository)라고 부릅니다.

GitHub(<https://GitHub.com>)에 접속하고 로그인을 합니다.

상단 오른쪽에 있는 [+]아이콘을 클릭한 후 [New repository] 메뉴를 선택합니다.

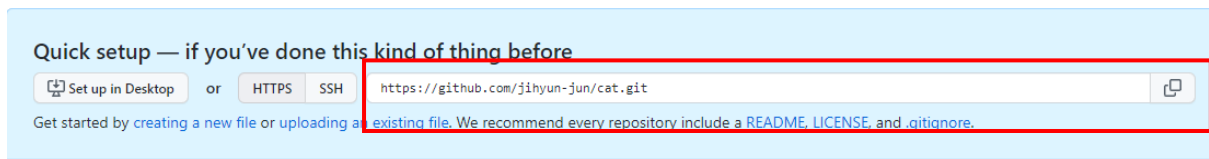


2) 어떤 저장소를 만들지 세부 항목을 작성하는 페이지가 나옵니다. [Repository name]에는 'cat'라고 기재하겠습니다. 이 이름으로 원격저장소를 만들겠다는 의미입니다. [Description]에는 생성할 원격저장소에 대한 간단한 설명을 작성하면 된다. 나머지 옵션은 별도로 변경하지 않고 [Create repository]버튼을 클릭합니다.

A screenshot of the 'Create a new repository' form on GitHub. The form has a light blue header with the title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' Below the header, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' section has a dropdown menu with 'jihyun-jun' selected. The 'Repository name' section has a text input field with 'cat' entered, followed by a green checkmark. Below these sections, there is a 'Description (optional)' section with a text input field containing 'it인을 위한 티셔츠 쇼핑물 오픈소스'. The 'Public/Private' section has two radio buttons: 'Public' (selected) and 'Private'. The 'Initialize this repository with:' section has a checkbox for 'Add a README file' and a dropdown for '.gitignore template' set to 'None'. The 'Choose a license' section has a dropdown for 'License' set to 'None'. At the bottom, there is a green button labeled 'Create repository'.

'cat'이라는 이름의 원격저장소를 만들었습니다. 화면을 보면 원격저장소 주소가 보일 겁니다. 예

를 들어, 'https://github.com/jihyun-jun/cat.git' 형태의 주소가 앞으로 공유되어질 원격주소가 됩니다.



또한 이 주소는 내 컴퓨터의 로컬 저장소와 연결할 때에도 사용합니다. 주소란 오른쪽 끝에 버튼을 누르면 원격 주소를 복사할 수 있습니다.

3.2. 원격저장소에 커밋 올리기

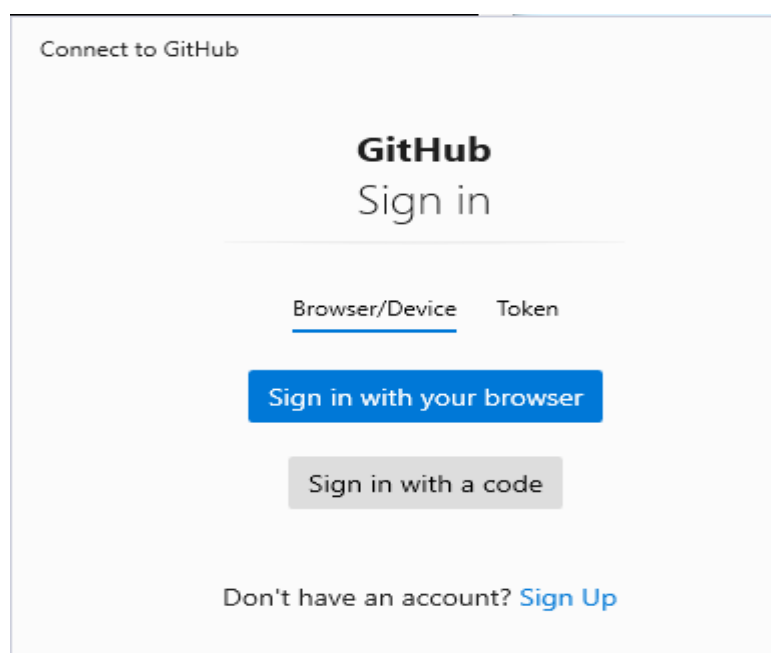
GitHub에 만든 [cat] 원격저장소 주소를 내 컴퓨터의 [cat-program] 로컬 저장소에 알려주고, 로컬 저장소에 만들었던 커밋들을 원격저장소에 올려 보겠습니다.

1) [cat-program] 폴더의 Git Bash로 들어옵니다. 'remote add origin' 명령어는 로컬 저장소에 원격 저장소 주소를 알려줍니다. 원격 저장소는 각자 만들어진 원격 저장소를 입력하면됩니다.

```
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git remote add origin https://github.com/jihyun-jun/cat.git
```

2) 이제 로컬저장소에 있는 커밋들을 push 명령어로 원격저장소에 올려 보겠습니다. 다음과 같이 명령어를 입력하고 'enter'를 누르면 GitHub의 로그인 창이 뜹니다. 각자의 GitHub 계정 정보를 입력하고 로그인을 합니다.

```
301-Te@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git push origin master
```



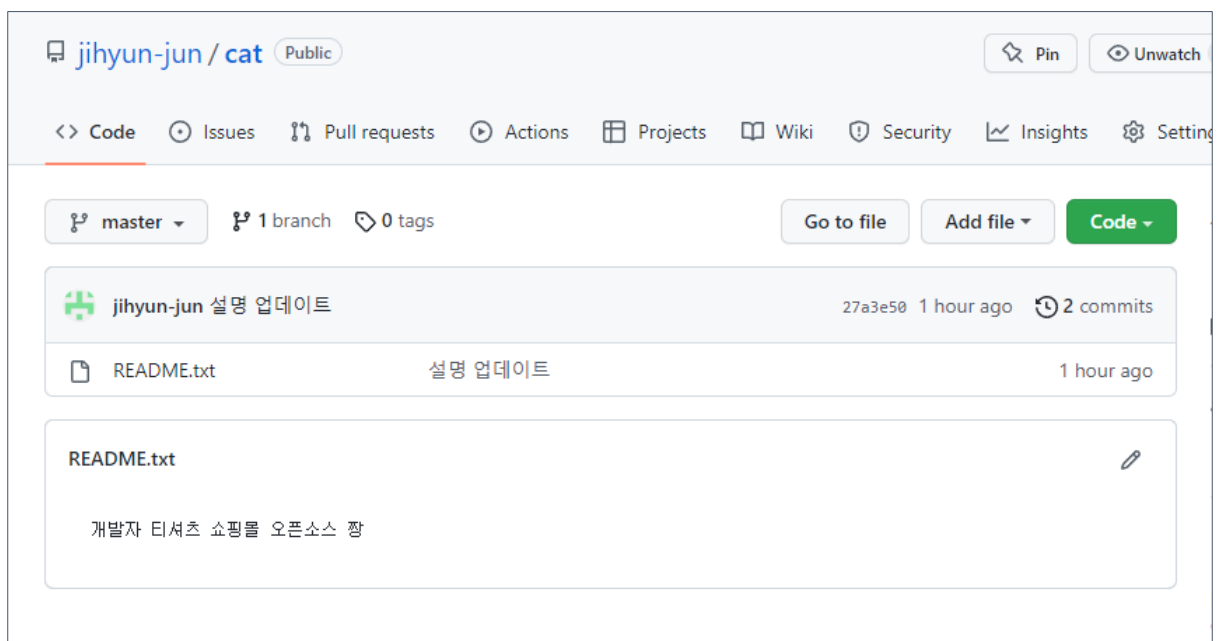
```

301-T@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 563 bytes | 281.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jihyun-jun/cat.git
* [new branch]      master -> master

```

3) 로그인에 성공하게 되면 입력했던 명령어가 실행되면서 100% 완료 텍스트가 나오면 성공입니다.

4) GitHub의 원격 저장소에서 확인을 해보겠습니다. GitHub의 “code” 메뉴를 눌러 실행해서 다음과 같이 화면이 뜨면 README.txt 파일이 잘 올라와 있는 것을 확인할 수 있습니다.



4. GitHub 원격장소의 커밋을 로컬장소에 내려받기

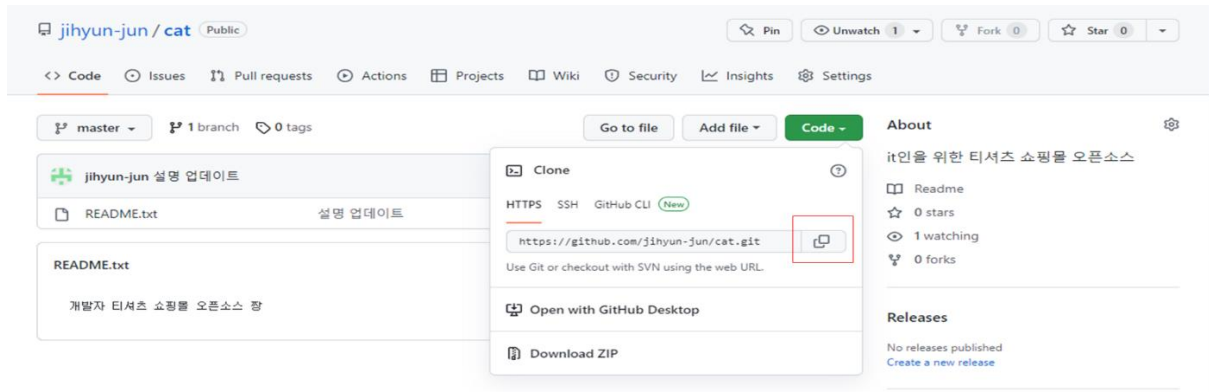
4.1. 원격저장소의 커밋을 로컬저장소에 내려받기

원격 저장소의 코드와 버전 전체를 내 컴퓨터로 내려받는 것을 클론(clone)이라고 합니다. 클론을 하면 최신 버전뿐만 아니라 이전 버전들과 원격저장소 주소 등이 내컴퓨터의 로컬저장소에 저장됩니다.

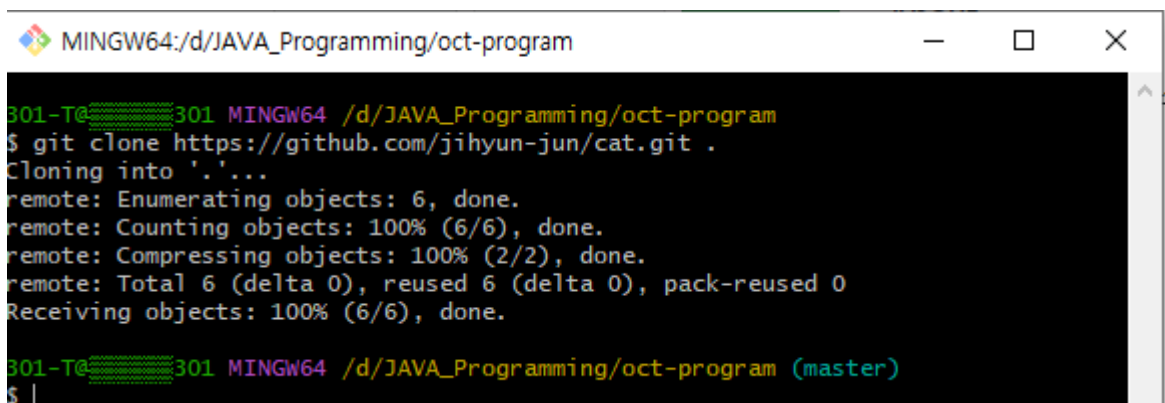
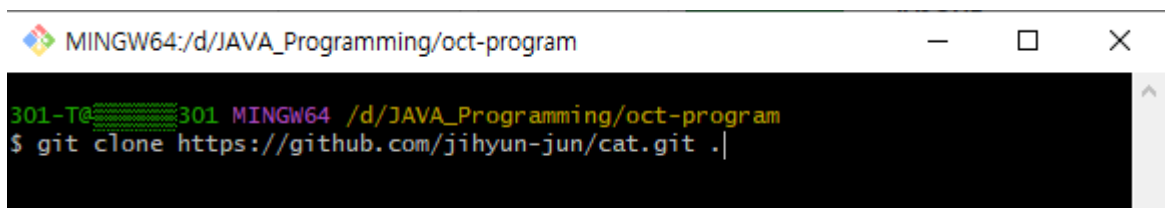
1) “D:\JAVA_Programming” 경로에 “oct-program”이라는 폴더를 만들어줍니다. 이 폴더를 문어의 로컬저장소로 지정하는 것입니다. 그리고, 고양이가 원격저장소에 올렸던 커밋을 이곳으로 내려받을 겁니다.

2) 방금 만든 "oct-program" 폴더에 마우스 오른쪽 버튼을 눌러 [Git Bash Here]를 클릭합니다.

3) clone 명령어 뒤에 원격저장소 주소를 입력하면 어느 원격저장소든 내 컴퓨터의 로컬저장소에 내려받을 수 있습니다. GitHub의 원격 저장소에 들어가서 [Clone or download] 버튼을 클릭하고 원격저장소 주소 오른쪽에 있는 버튼을 클릭해서 원격저장소 주소를 복사합니다. [Download ZIP]으로도 소스코드를 똑같이 받을 수 있지만 그러면 원격저장소와 버전 정보가 제외되니 꼭 클론을 통해 받아야합니다.



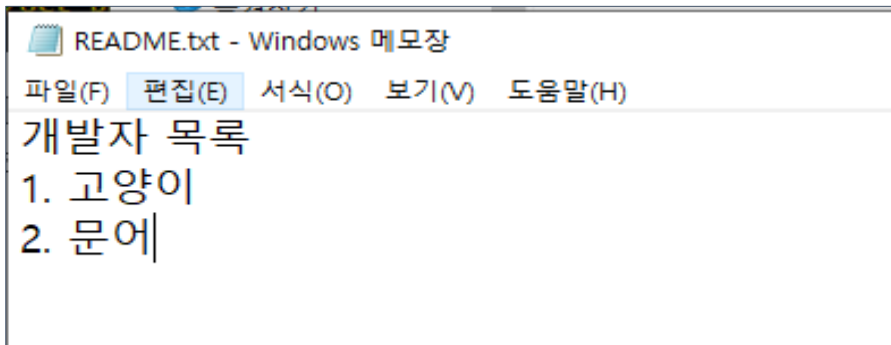
Git clone 명령어 뒤에 원격저장소 주소를 적고(방금 복사한 것을 붙여넣고) 한 칸 띄고 마침표를 찍어줍니다. 마침표는 현재 폴더 [oct-program]에 받으라는 뜻입니다. 점을 찍지 않으면 [oct-program] 안에 [cat] 폴더가 생깁니다. 원격저장소 뒤에 한 칸 띄고 마침표를 넣는 것을 빼먹지 말아야 합니다.



4) [oct-program] 폴더에 README.txt 파일과 [.git] 폴더가 보이면 성공입니다. 클론을 하면 이렇게 로컬저장소가 자동으로 생깁니다. 고양이가 Git 초기화를 해서 만들었던 로컬저장소를 받아온 겁니다.

5) [oct-program] 폴더에 있는 README.txt 파일을 열어보면 원격 장소에 커밋한 것과 동일한 내용이 저장되어 있는 것을 확인할 수 있습니다.

6) 여기서 새로 커밋을 만들어 올려보겠습니다. README.txt 파일을 아래처럼 수정하고 저장합니다.



7) Git bash에서 아래 세 개의 명령어를 입력하여 실행합니다.

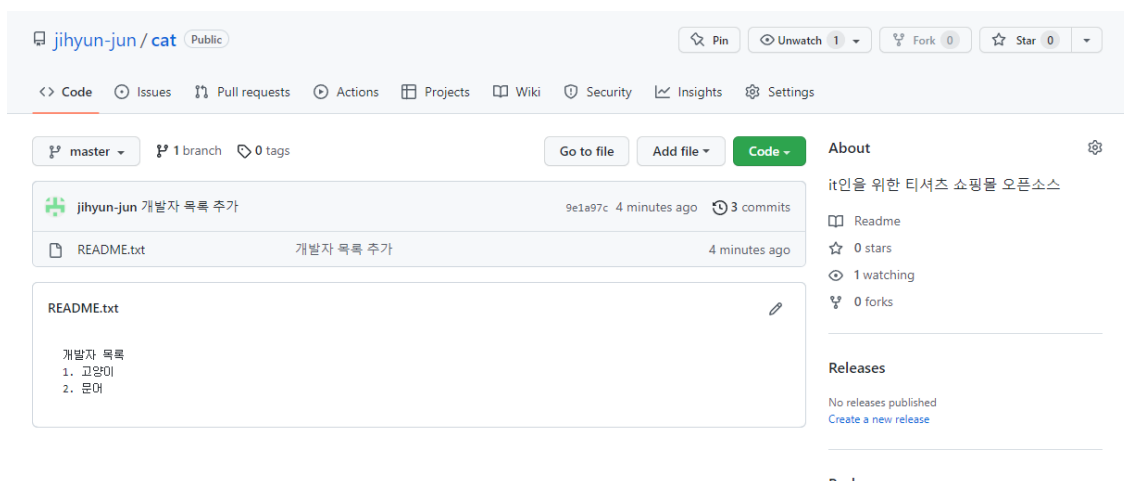
```
301-T@301 MINGW64 /d/JAVA_Programming/oct-program (master)
$ git add README.txt

301-T@301 MINGW64 /d/JAVA_Programming/oct-program (master)
$ git commit -m "개발자 목록 추가"
[master 9e1a97c] 개발자 목록 추가
1 file changed, 3 insertions(+), 1 deletion(-)

301-T@301 MINGW64 /d/JAVA_Programming/oct-program (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jihyun-jun/cat.git
27a3e50..9e1a97c master -> master

301-T@301 MINGW64 /d/JAVA_Programming/oct-program (master)
$ |
```

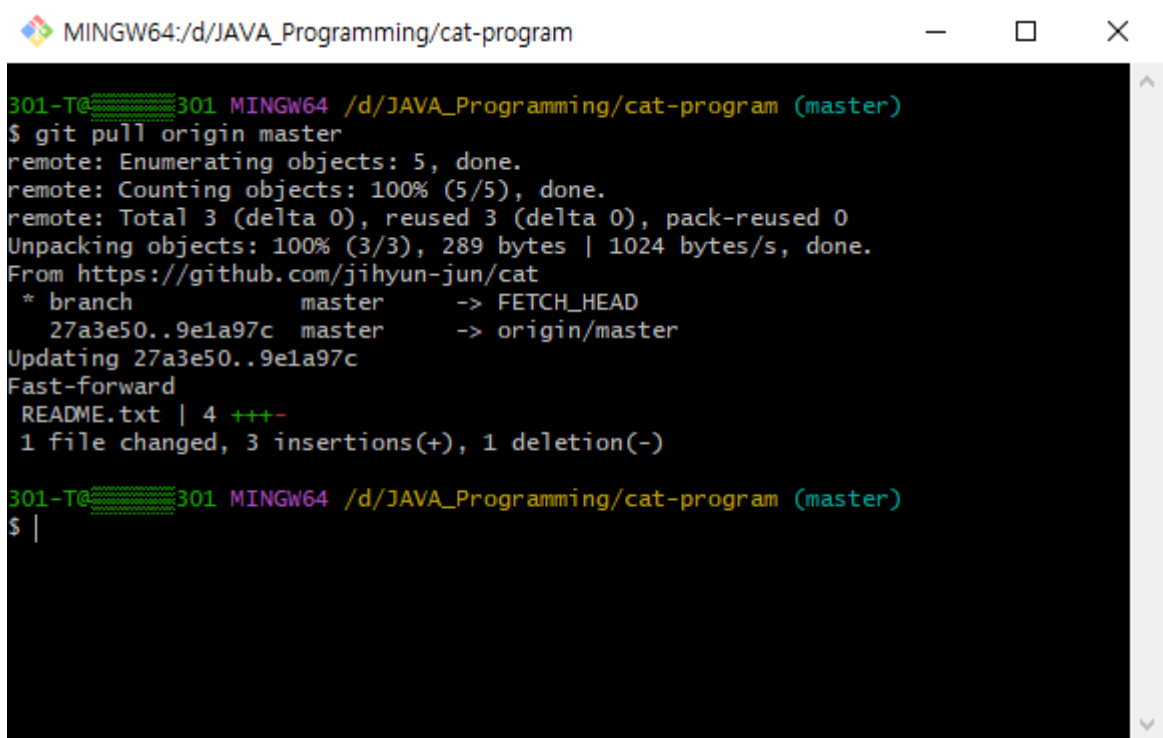
8) GitHub의 원격저장소에 들어가서 새로고침을 해보면 우리가 push한 '개발자 목록 추가' 커밋이 되어 있는 것을 확인할 수 있습니다.



4.2 원격저장소의 새로운 커밋을 로컬 저장소에 갱신하기

앞서 [oct-program] 폴더의 README.txt 파일을 갱신하여 원격저장소에 올렸습니다. 하지만 아직 [cat-program]에는 갱신된 README.txt 파일을 갱신하여 내려받지 못했습니다. 아래 과정을 통해 원격저장소에 커밋된 내용을 로컬 저장소에 내려받아 갱신 작업을 해 보겠습니다.

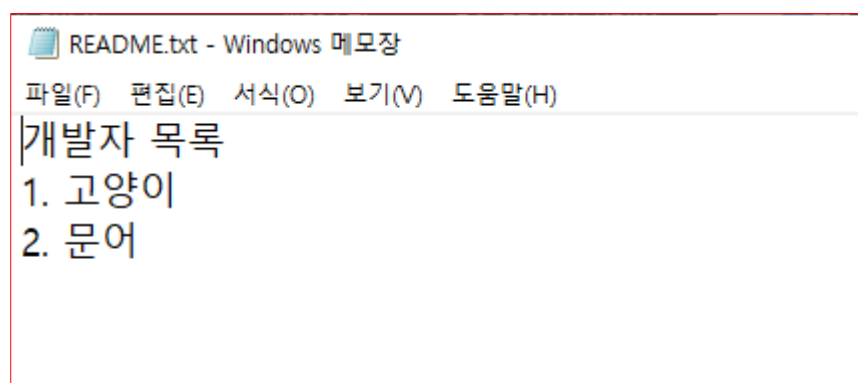
- 1) [cat-program] 폴더로 가서 README.txt 파일을 엽니다. 세 번째 커밋이 반영되지 않은 옛날의 두 번째 '설명 업데이트' 커밋의 상태에 머물어 있음을 알 수 있습니다.
- 2) [cat-program] 폴더에서 오른쪽 버튼을 눌러 [Git Bash Here]를 선택합니다.
- 3) 아래 명령어를 입력하여 실행을 합니다.(원격저장소에 새로운 커밋이 있다면 그걸 내 로컬 저장소에 받아오라는 명령어입니다.)



```
MINGW64:/d/JAVA_Programming/cat-program
301-T@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 289 bytes | 1024 bytes/s, done.
From https://github.com/jihyun-jun/cat
* branch      master      -> FETCH_HEAD
   27a3e50..9e1a97c master    -> origin/master
Updating 27a3e50..9e1a97c
Fast-forward
 README.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

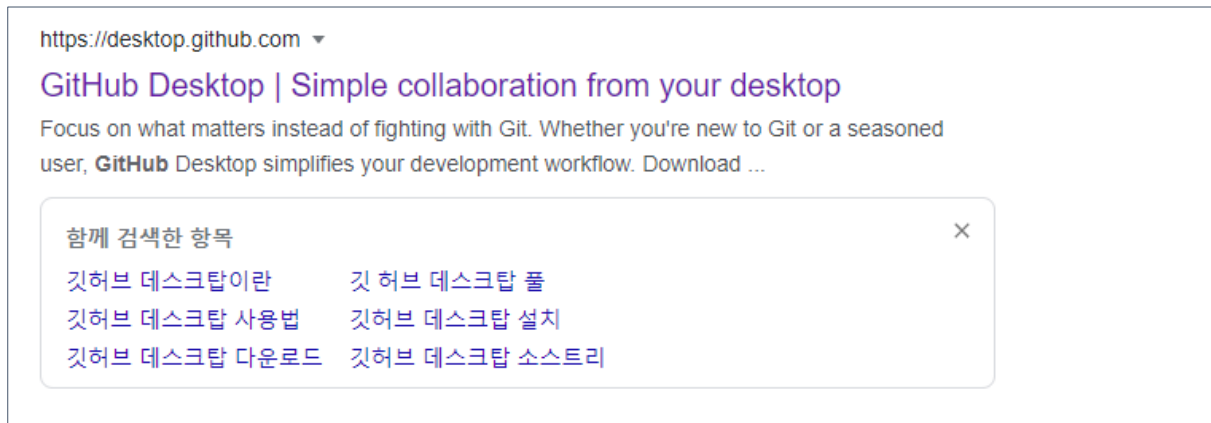
301-T@301 MINGW64 /d/JAVA_Programming/cat-program (master)
$ |
```

- 4) 1 file changed 메시지가 나오면 성공입니다. [cat-program] 폴더에서 README.txt 파일을 다시 열어서 내용을 확인해 봅니다.

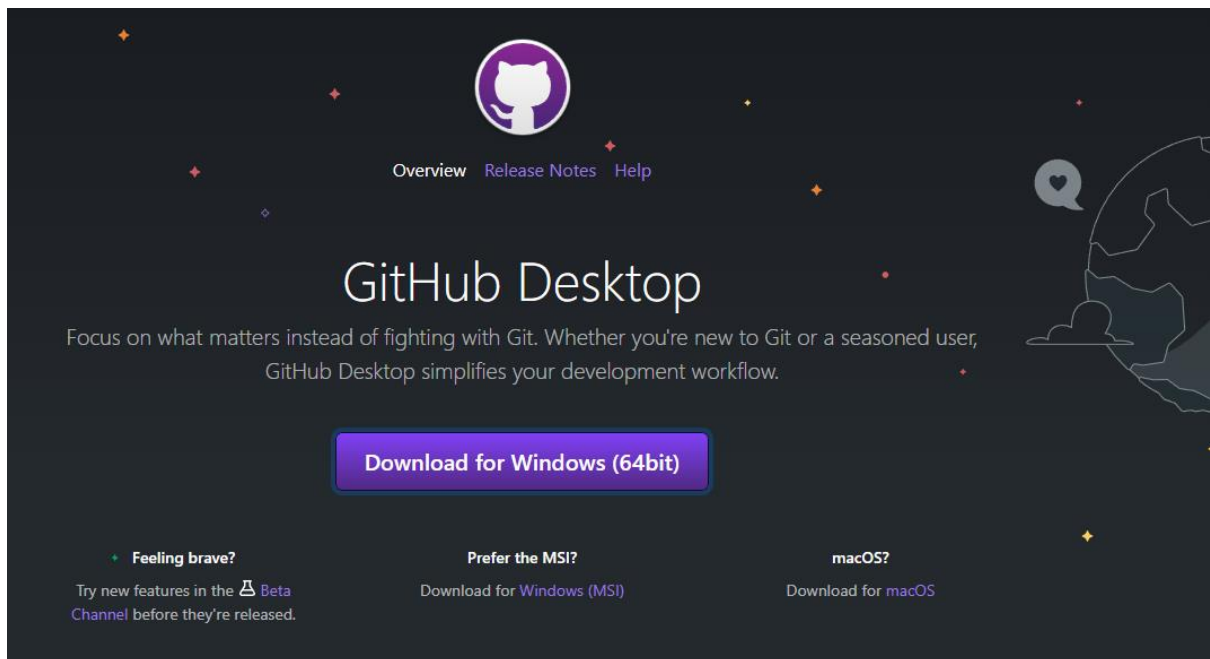


5. Git Hub Desktop 설치하기

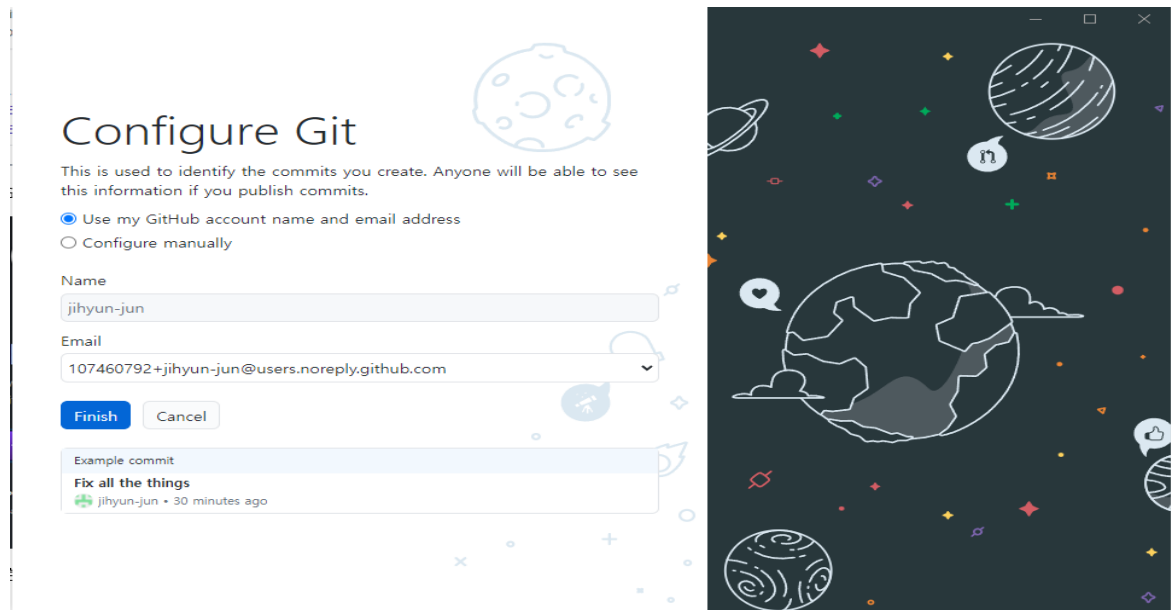
- 1) 구글 검색창에 "깃 허브 데스크 탑" 검색
- 2) 상단에 있는 desktop.github.com 클릭



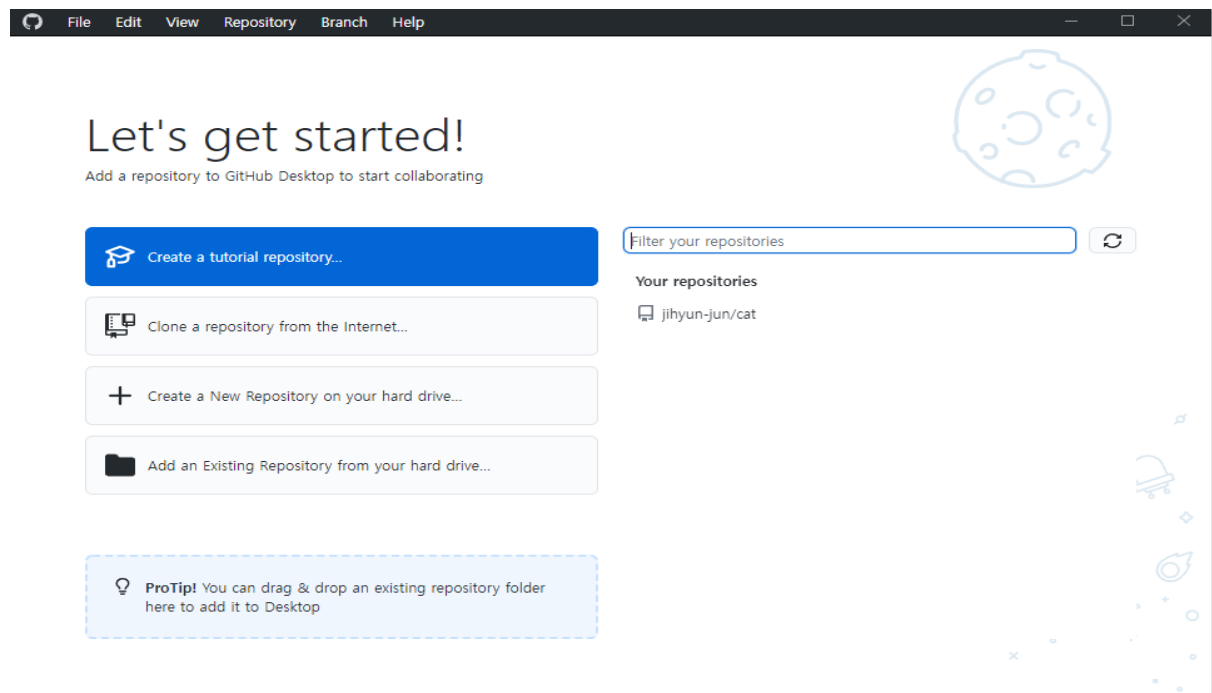
- 3) URL 접속 후 메인 화면이 나오면 GitHub Desktop 프로그램을 다운로드 받는다.



- 4) 다운 받은 GitHub Desktop를 설치한다.



5) 설치 완료 후 데스크탑을 실행한다.



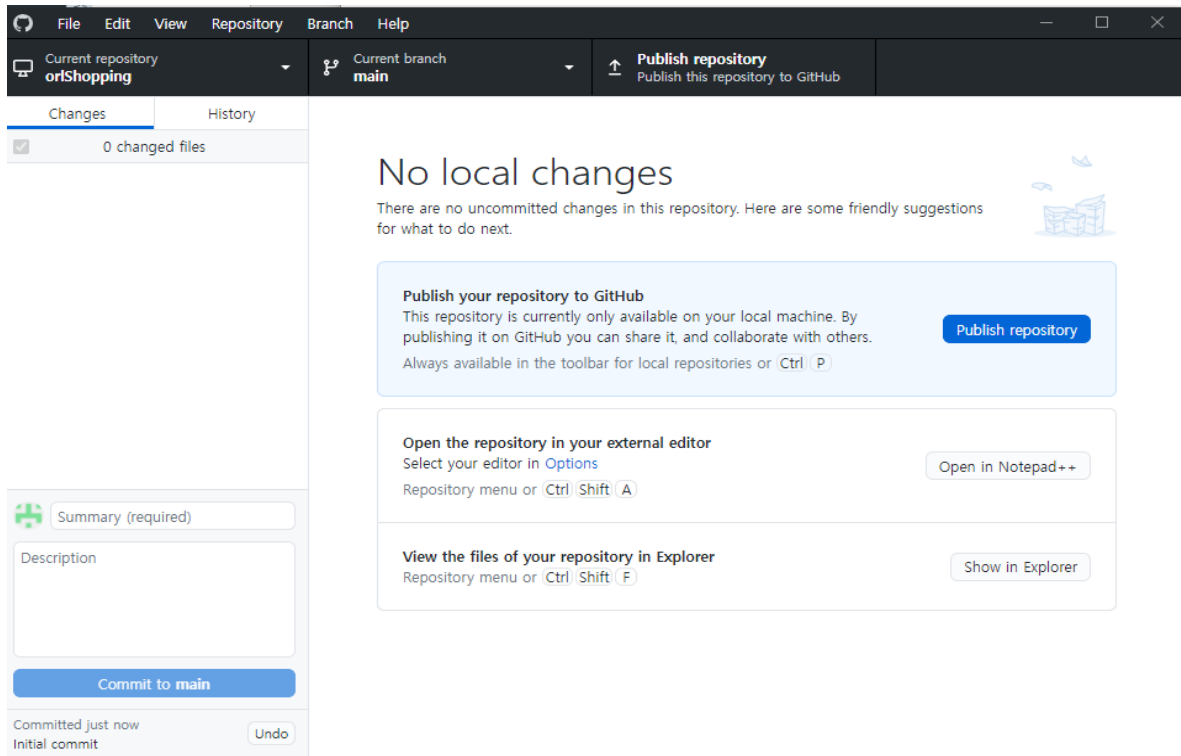
6) Create a New Repository on your Hard Drive를 클릭한다.

7) 각 항목을 기입한 후 "Create Repository"를 클릭한다.

- Name : 프로젝트명(LocalPath에 만들어질 폴더명)

- Description : 프로젝트 설명

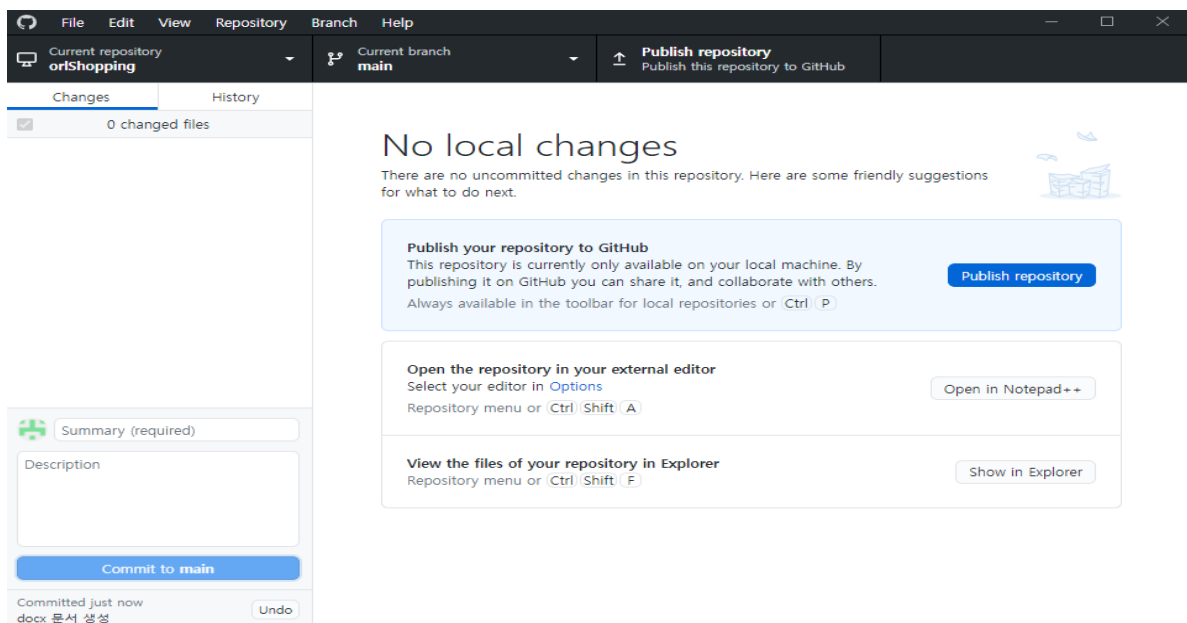
- LocalPath : 깃으로 관리할 폴더(문서 파일 있거나 있을 곳)



위와 같은 화면이 나오면 폴더 깃관리가 지정 완료되어짐.

8) 깃 관리 지정된 폴더내(local-Path) 문서 파일 생성 후 깃 데스크톱 프로그램 확인하기 위하여 만들어진 폴더내에 문서를 작성해서 저장하게 되면 Changes에 방금 만들어진 파일이 보일것이다.

하단의 정보 입력 란에 버전 명과 설명글을 적고 "Commit to main"클릭한다. 클릭 후 화면이 바뀌는데, changes에는 모든 항목이 사라져 있고, 오른쪽에는 publish repository 버튼이 하나 만들어진 것을 확인할 수 있다.



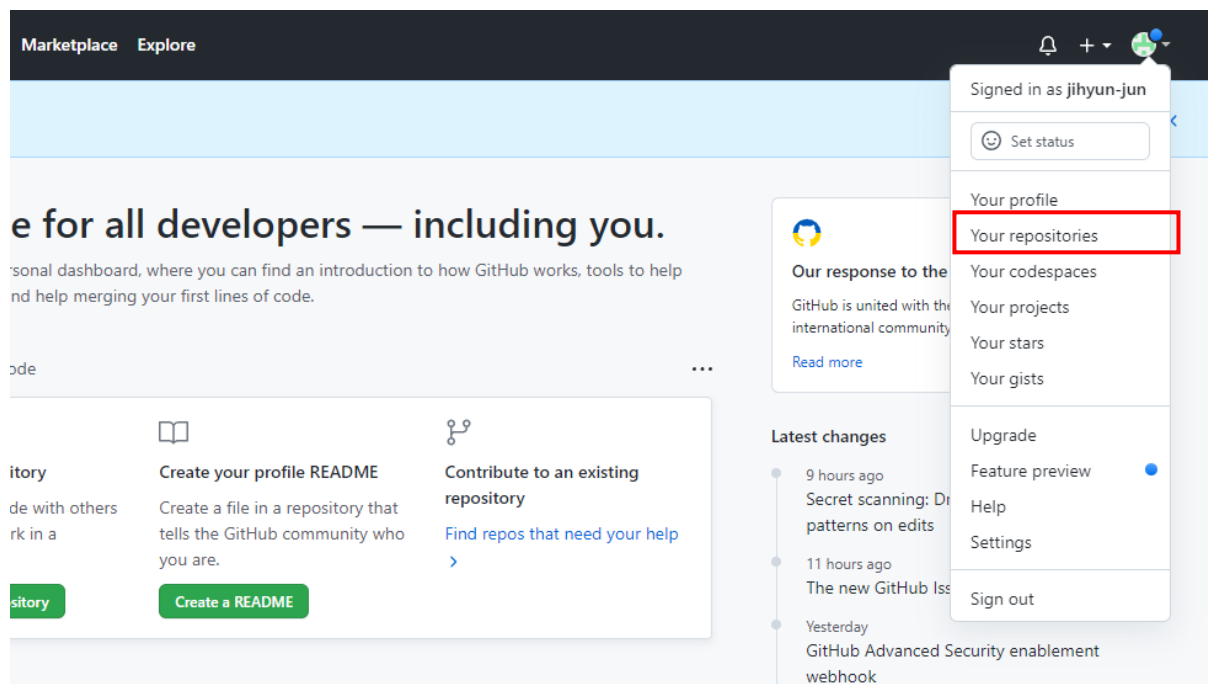
9) 깃 허브에 publish하기 위해 해당 버튼을 클릭하면 깃 허브에 폴더를 업로드하고 다른 데스크톱에서 얼마든지 깃을 실행하여 문서편집을 이어갈 수 있다.

10) Name과 Description을 다시 적으라고 나오는데 온라인상의 Project 폴더 명과 설명글이라고 생각하면 된다. 하단에 keep this code private 체크시 해당 프로젝트는 본인만 확인 가능

6. GitHub에 소스 올려 연동하기

1) 이클립스에서 깃허브를 사용하기 위해서 깃허브 홈페이지에서 저장소를 구축해야 합니다.

GitHub에 로그인한 후에 오른쪽 상단의 사람 모양을 클릭하여 "Your Repositories"를 클릭합니다.

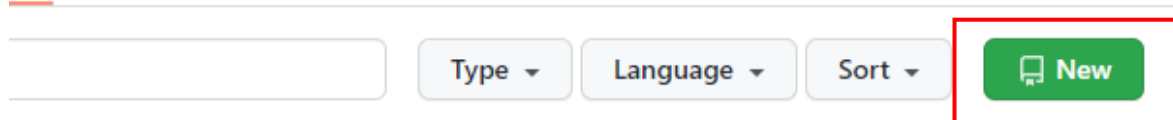


2) "New"를 클릭하여 저장소를 생성합니다.

- Add a README file : 저장소를 소개할 텍스트를 입력할 수 있는 파일을 생성

- Add .gitignore : 깃허브에서 추적할 수 없는(연동 X)파일의 리스트를 입력할 수 있는 파일을 생성합니다.


ies Projects Packages Stars



Create a new repository

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

Owner *

 jihyun-jun ▾

Repository name *

sample ✓

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

Description (optional)



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.](#)

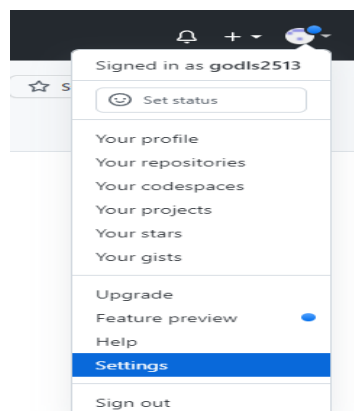
License: None ▾

 You are creating a public repository in your personal account.

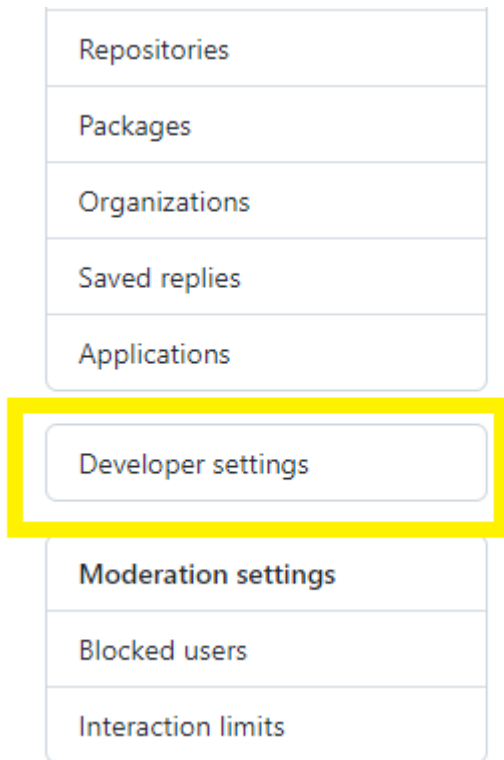
Create repository

3) 이클립스와 GitHub를 연결해주어야 하는데 2021년 8월부터 git 작업을 인증할 때 계정 암호를 더 이상 승인하지 않는다고 합니다. 이를 해결하기 위해 인증 token를 만들고 그것을 이용해 이클립스와 연동을 해야 합니다.

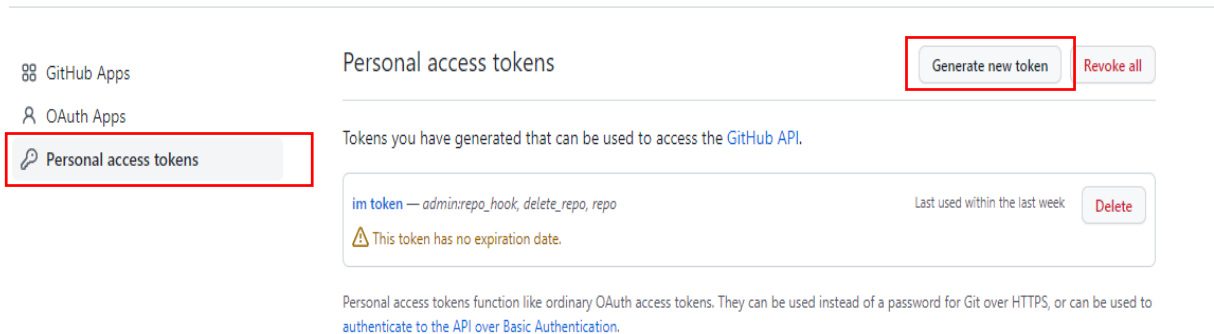
인증 token를 생성하기 위해 우측상단의 프로필 로고에서 settings를 누른다.



4) 좌측 메뉴에서 Developer settings를 누른다.



5) 좌측 메뉴에 Personal access tokens -> 우측 상단에 Generate new token 클릭합니다.



6) 아래 그림과 같이 셋팅 후 하단에 Generate token을 누른다. Note는 짓고 싶은 이름으로 하면 되고, Expiration의 경우 유효기간을 정할 수 있는 No expiration으로 유효기간을 없음으로 하면 된다.

GitHub Apps

OAuth Apps

Personal access tokens

New personal access token

Personal access tokens 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.

Note

my token

What's this token for?

Expiration *

No expiration

The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

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 type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo:deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input 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
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete:repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage:runners:enterprise	Manage enterprise runners and runner groups
<input type="checkbox"/> manage:billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys (Developer Preview)
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys

Generate token

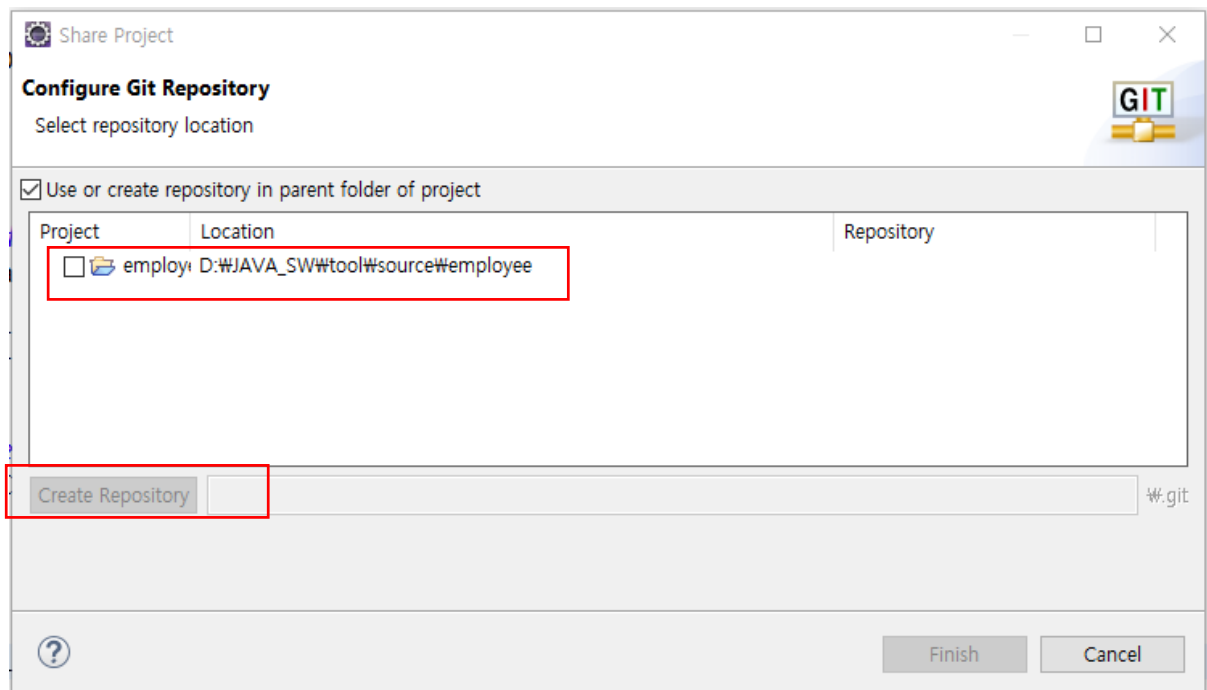
Cancel

7) 생성된 token을 복사해서 잃어버리지 않게 보관한다. (계속 사용이 가능함.)

8) 이클립스를 열고 연동할 프로젝트 우클릭>Team>Share Project를 클릭합니다.

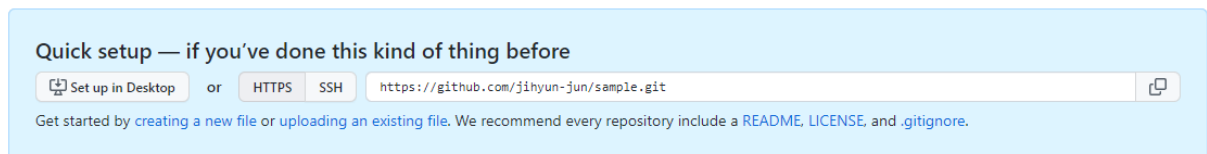
9) "Use or create repository in parent folder of project"에 체크 해체가 되어 있을텐데 체크를 해 준다. 그러면 아래와 같은 화면으로 변경이 되는데 네모 부분의 location의 내용을 마우스 좌클릭

한 다음 create Repository를 누른 다음 밑에 Finish를 누른다.



7. 이클립스에서 연동된 프로젝트를 GitHub에 연동하기

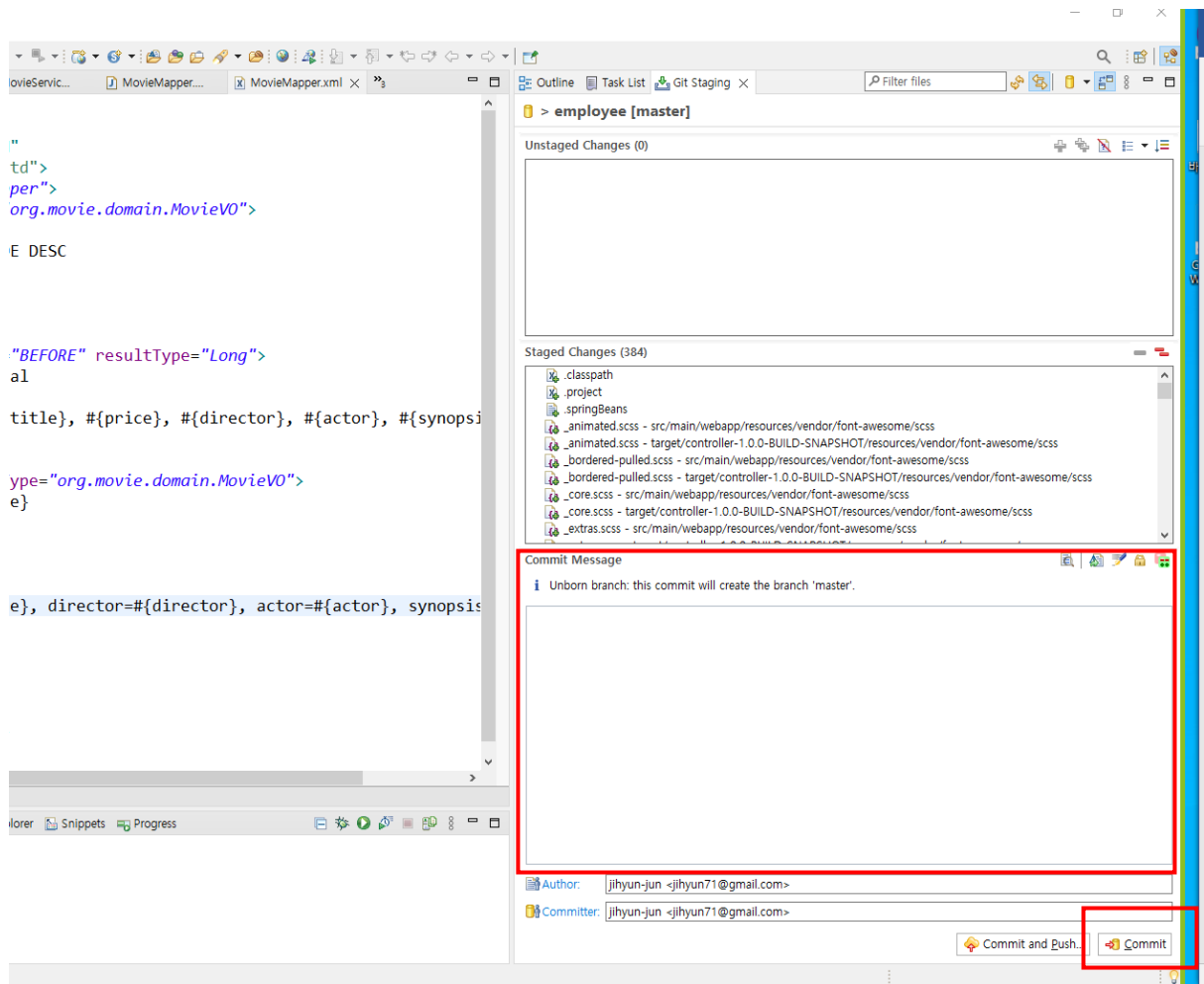
1) Github에서 처음 생성했는 Repository의 주소를 복사한다.



2) 이클립스에서 프로젝트 우클릭->Team->Add to Index를 누른다.

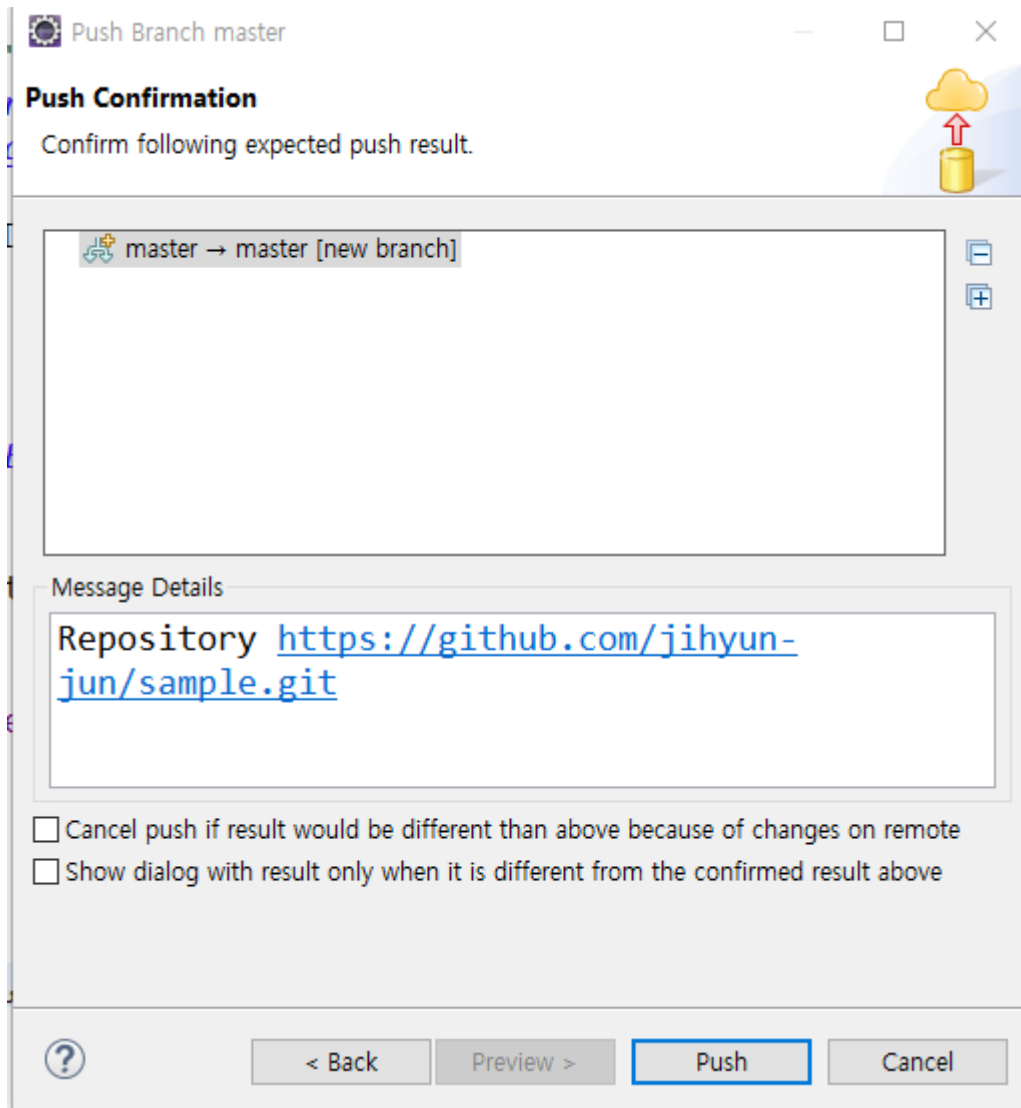
3) 다시 프로젝트 우클릭->Team->Commit를 누른다.

4) Commit를 누르면 다음과 같이 항목이 뜨고 Commit Message 입력 후 우측 제일 하단에 Commit을 누른다.

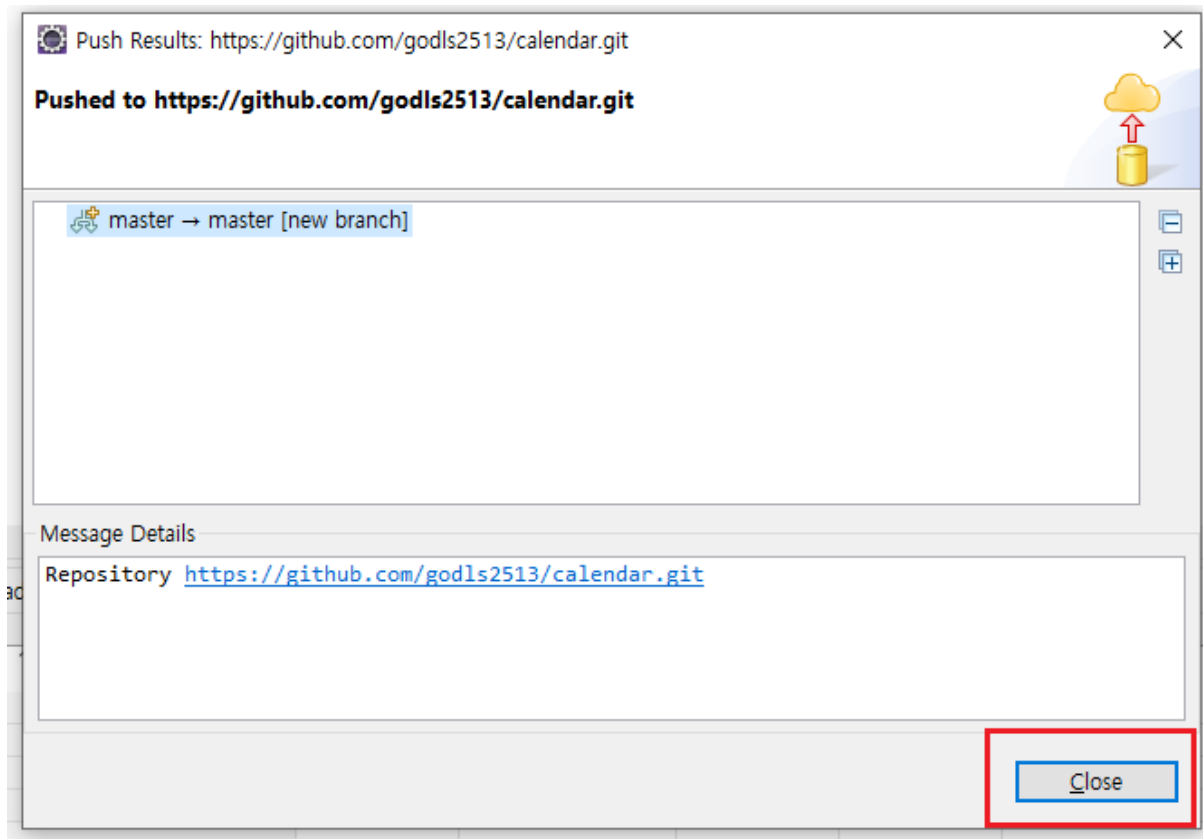


5) 이후 프로젝트 우클릭->Team-> Push Branch 'master'... 클릭한다.

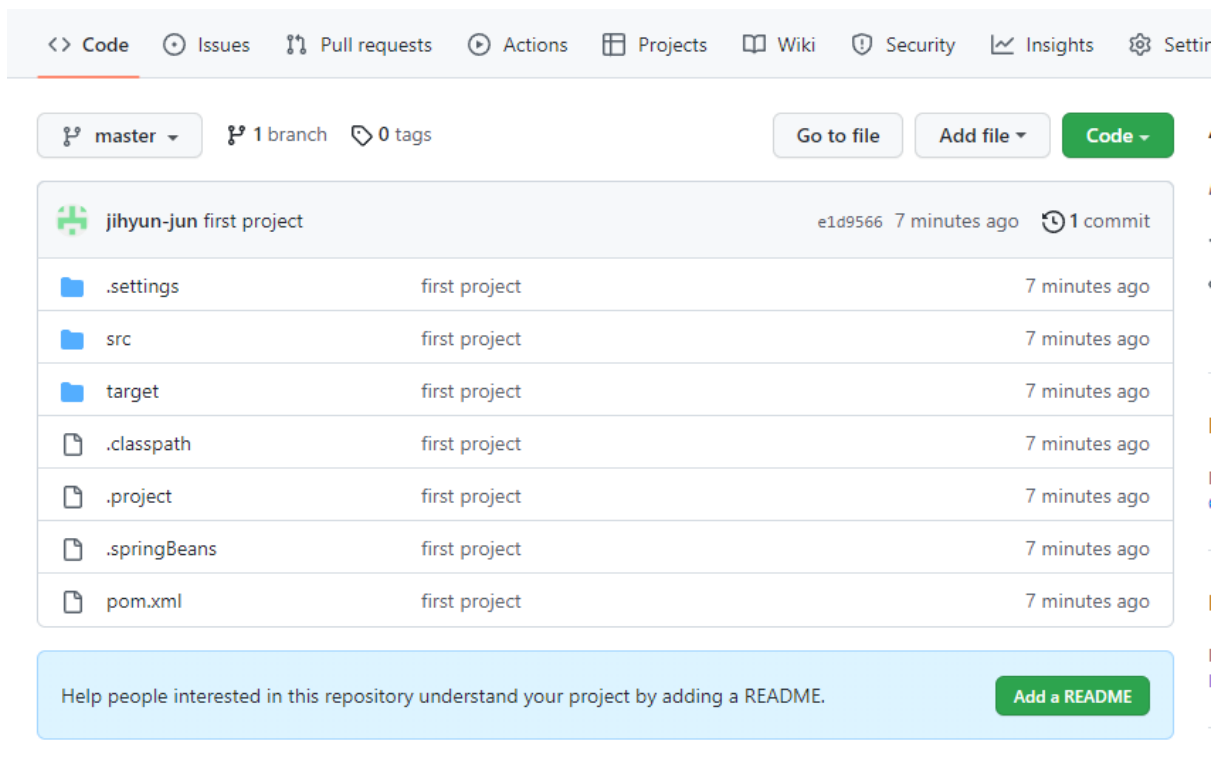
- URI에는 github에 생성된 repository 주소 복사한 것을 붙여넣기 한다.
- user에는 github username과 password는 앞에서 생성한 token을 복사해 붙여넣기 한다.
- 그리고 Stor in Secure Store를 체크해 다음에도 Password를 기억할 수 있도록 한다.
- 다 입력이 되었으면 Prwview 버튼을 누른다.



7) close 버튼을 누릅니다.

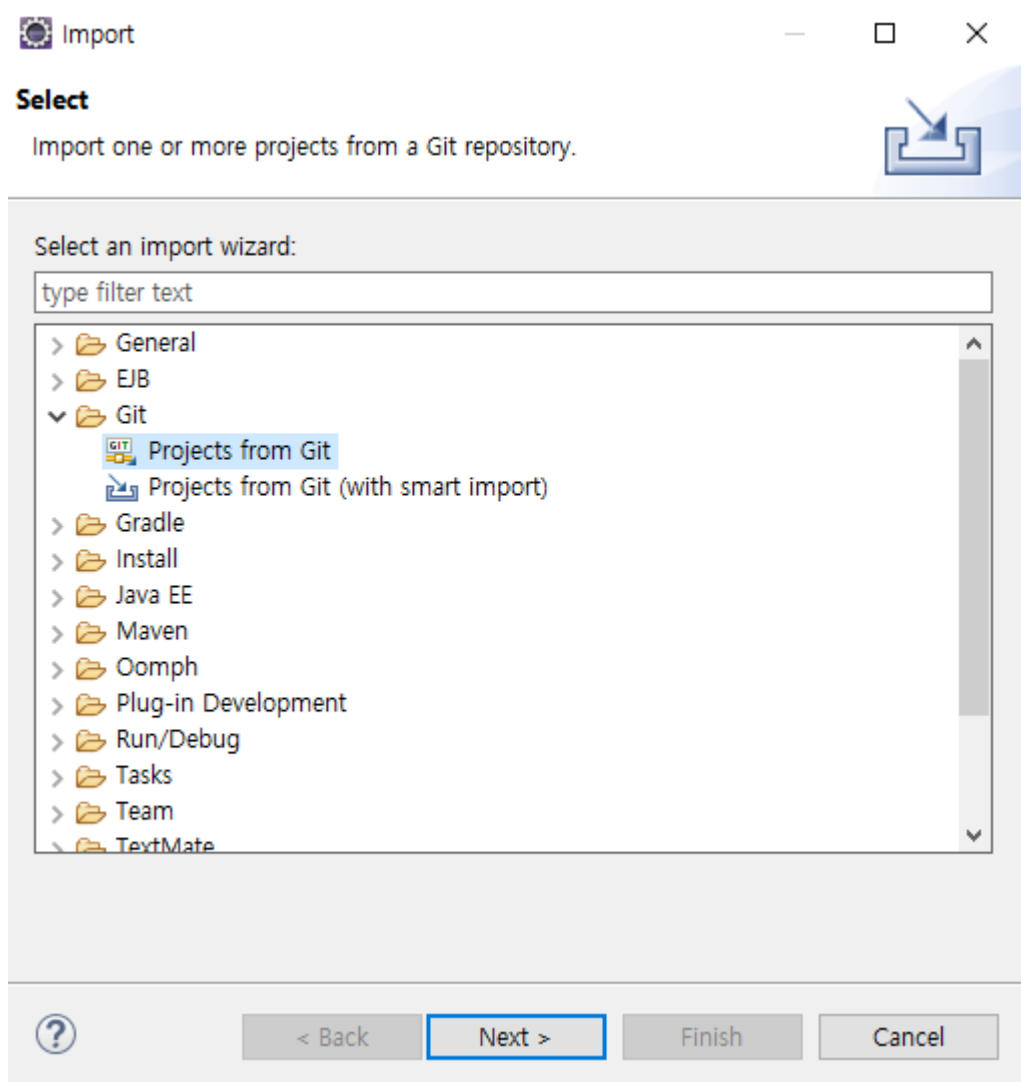


8) github에 들어가서 push가 잘 되었는지 확인한다.

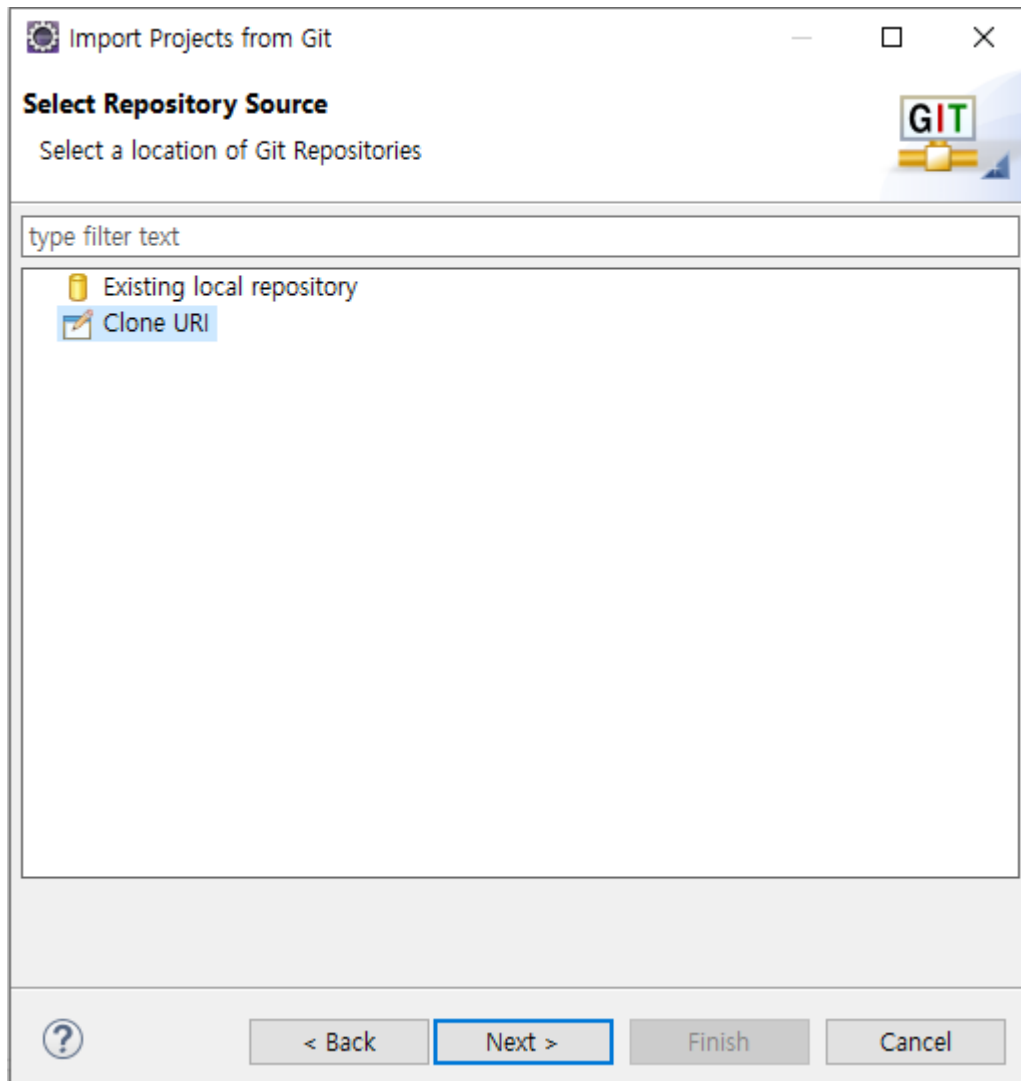


8. 이클립스로 Github에 있는 프로젝트 가져오기

1) File>import를 실행한 후 "Git" 폴더 밑에 "Projects from Git"를 선택한 후 "next"를 누릅니다.



2) "Clone URI"를 선택합니다.



3) GitHub 화면에서 복사한 URL 주소를 URI에 붙여넣기를 합니다. 자동으로 Host와 Repository path가 채워집니다.

Local Destination

Configure the local storage location for sample.



Destination

Directory: D:\JAVA_SW\tool\git_source\sample

Browse

Initial branch: master



☐ Clone submodules

Configuration

Remote name: origin

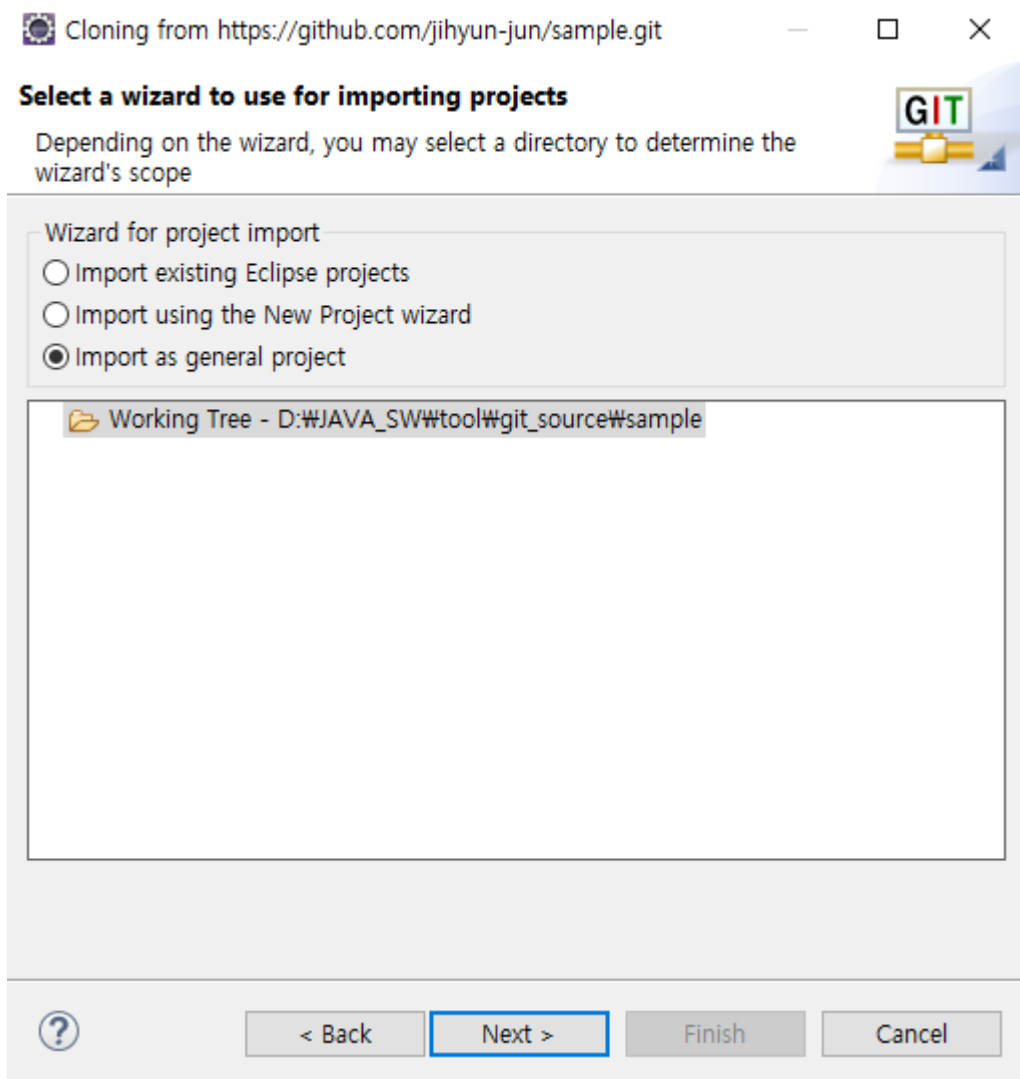


< Back

Next >

Finish

Cancel



만약 프로젝트를 가져올 때 위의처럼 "import as general project"로 선택하고 가져오기 했을 때 오류가 발생이 되면 "import existing Eclipse projects"를 클릭하여 프로젝트를 가져오면 됩니다.