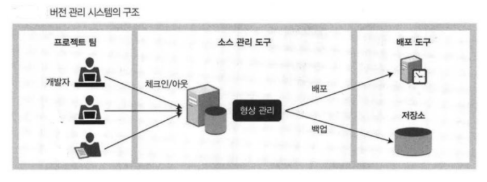
**소스 형상 관리 정의**

소프트웨어 형상관리는 Software Configuration Management, 줄여서 SCM라는 단어를 쓰기도 하는데, SW개발 및 유지보수 과정에서 발생하는 소스코드, 문서, 인터페이스 등 각종 결과물에 대해 형상을 만들고, 이들 형상에 대한 변경을 체계적으로 관리, 제어하기 위한 활동입니다. 단순히 말하자면 프로젝트를 진행하면서 생성하는 소스코드를 CVS나 SVN, 또는 GIT와 같은 버전 관리 시스템을 이용하는 것을 말합니다. 다수의 개발자가 프로젝트에서 동일한 기능을 동시에 개발한다고 할 때, 작성된 소스 코드와 변경사항을 확인하고, 수정하는 협업을 도와주는 시스템이라고 할 수 있습니다.

형상관리는 일반적으로 버전 관리 (version control, revision control), 소스 관리 (source control), 소스 코드 관리 (source code management, SCM)와 동일한 의미로 사용됩니다. 즉, 동일한 정보(프로그램)에 대한 여러 버전을 관리하는 것으로, 소프트웨어 공학에서는 팀 단위로 개발 중인 소스 코드나 청사진(설계도) 등 디지털 문서의 작업 단계별 버전을 관리하는 작업으로 정의됩니다. 소스 형상관리 툴의 대략적인 구조는 아래와 같습니다.



**소스 형상관리는 왜 해야할까?**

방대한 양의 개발을 혼자서 하기에는 무척이나 힘이듭니다. 고로 개발자들간의 협업은 매우 중요하죠. 공동으로 개발작업을 하다보면 서로가 수정한 소스를 최신 버전으로 공유하여 개발을 진행하여야 합니다. 간단한 프로그램이라면 그냥 카카오톡이나 메일로 소스코드를 주고받으면서 할수도 있겠지만 이것은 소스의 양이 많아지고 페이지가 많아질수록 비효율적이게 됩니다. 형상관리툴을 활용하여 프로그램 소스라면 특정 저장소(레파지토리)에 저장해놓고 본인이 수정한 최신 소스를 업로드하면서 다른 사람의 최신버전을 내려받아 개발을 진행하는 것이 필요한데, 이를 형상관리라고 합니다. 소스의 형상(모양)을 관리한다는 것 말 그대로 전체적인 틀을 유지한다는 것입니다.

**소스 형상관리 툴 사용시 얻는 장점**

1. 소스 코드를 프로젝트 팀원 및 관계자들과 공유할 수 있다.

2. 소스 코드의 변경 이력을 관리할 수 있다.

3. 서버나 클라이언트에 배포할 때에도 유용하게 사용된다.

4. 여러 사람이 동일한 소스 코드를 공유해서 개발할 수 있으며 소스 코드를 공유할 때 생기는 버전 충돌 문제를 해결할 수 있다.

5. 장애 혹은 기능상 필요할 때 이전 버전으로 소프트웨어를 원상복구할 수 있다.

6. 동일한 소프트웨어를 여러 개의 버전으로 분기해서 개발할 필요가 잇는 경우에 유용하게 사용된다

**소스형상관리 툴**

**Git**

'깃(Git)'은 2005년 리눅스를 만든 리누스 토발즈와 주니오 하마노가 개발한 분산형 버전관리 시스템입니다. 어떤 코드를 누가 수정했는지 기록, 추적을 할 수 있는 오픈 소스 소프트웨어로 수천명의 사람들이 이용해도 안정적이고, 속도가 빠르다는 장점이 있습니다.

**Git Hub**

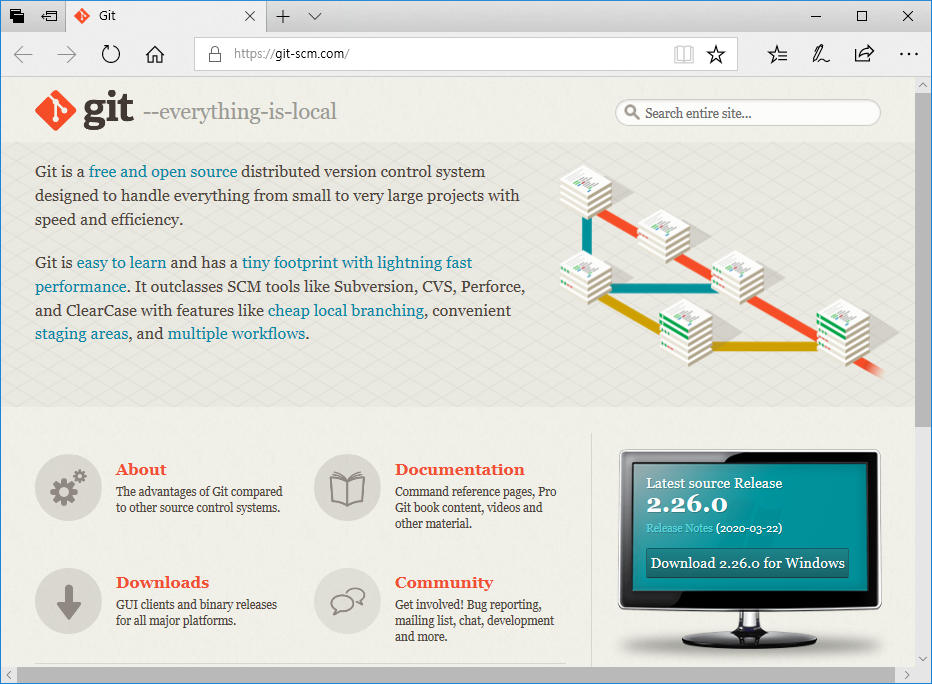
깃허브는 개발자들끼리 소스를 공유할 수 있는 오픈소스 페이지(개발자 스스로 본인의 소스를 다른사람들에게 공개하여 필요한 정보를 공유할 수 있는 자유로운 공간) , SNS기능, 블로그 기능까지 추가하는 등 소스형상관리기능에서 개발 플랫폼으로 점점 진화하고 있습니다. 개발자들이 굉장히 많이 사용하고 친숙한 형상관리툴입니다.

**Git Extension**

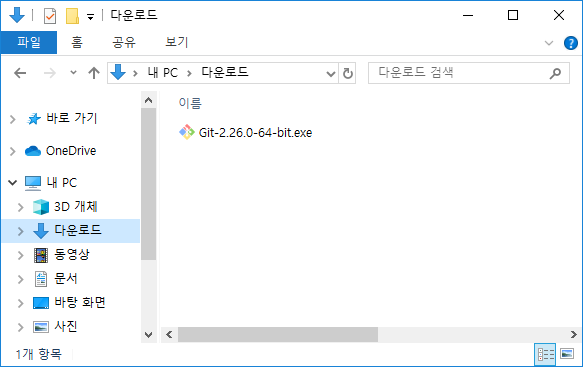
'깃'은 Git bash 명령어를 입력하면서 이용해야되는 불편함이 있었습니다. 이 불편함을 더욱 편리하게 이용할 수 있게 만든게 바로 'Git Extension'입니다.

**GIT 설치하기**

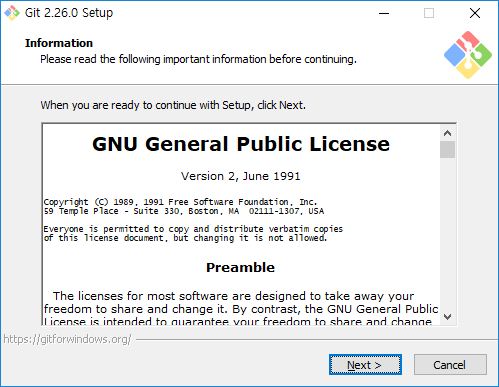
1. <https://git-scm.com/> 링크 페이지에 들어가서 자신의 OS에 맞는 Git 설치버전을 다운받는다.



1. 다운로드 완료한 Set-Up File을 실행합니다.

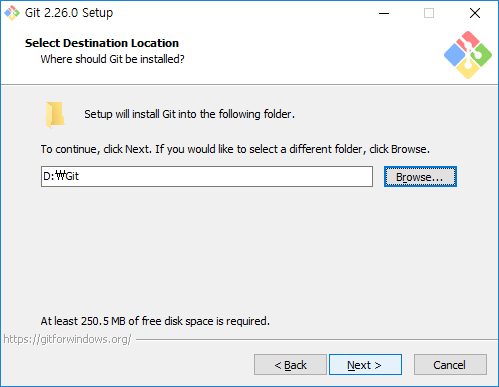


1. 약관을 읽어주고 Next를 눌러줍니다.

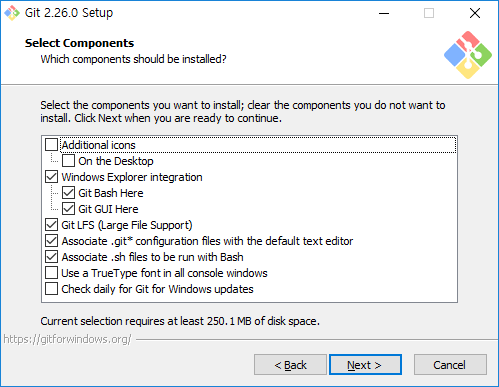


**<그림-4 GIT 설치 약관>**

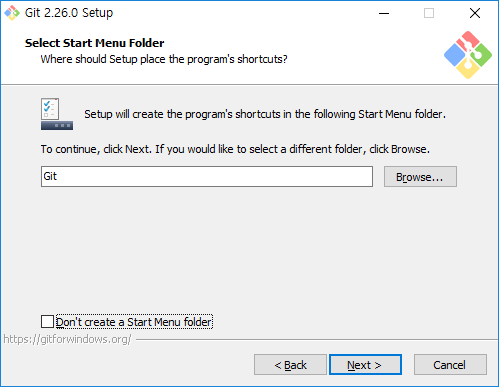
1. 설치경로를 선택하고 Next를 눌러줍니다.



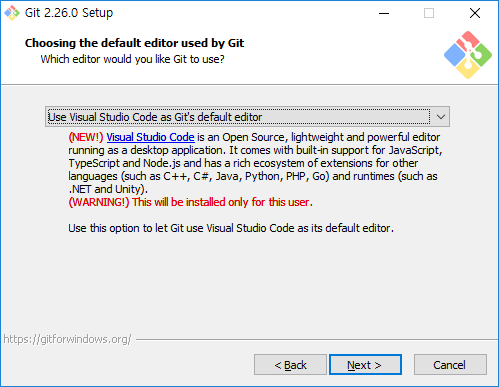
1. 설치할 Component들을 선택합니다. 저는 기본으로 선택되어 있는것 이외에는 추가로 설치하지않고 Next를 눌러 진행하겠습니다.



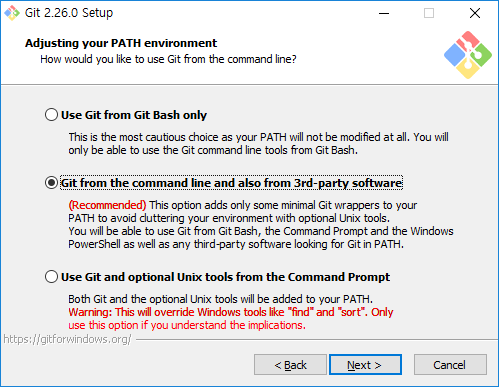
1. 시작 메뉴에 폴더를 만듭니다. 시작메뉴에 폴더를 추가하고 싶지 않다면 Don't create a Start Menu folder 체크박스를 클릭하고 Next를 눌러주세요.



1. 깃의 기본에디터를 설정합니다. 설정해주고 Next~



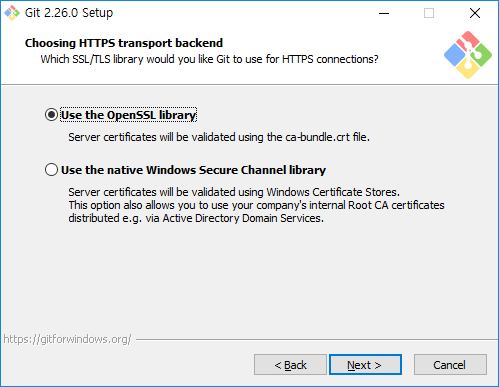
1. Git커맨드의 설정을 하는 부분입니다. 저는 기본으로 되어있는 Use Git from the Windows Command Prompt를 선택하고 Next를 하겠습니다.



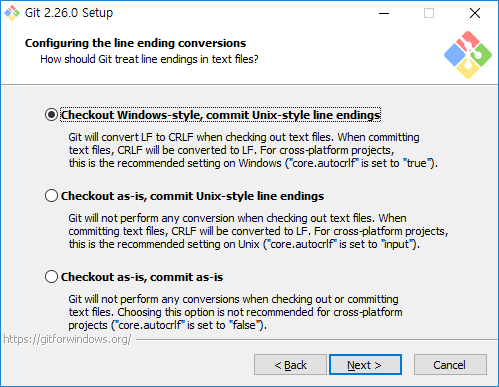
1. 디폴트로 되어있는 Use the OpenSSL library를 선택해주고 Next를 눌러줍시다.

Use the OpenSSL library : OpenSSL 라이브러리 사용하며 서버인증서는 ca-bundle.crt 파일을 사용하여 유효성 검사

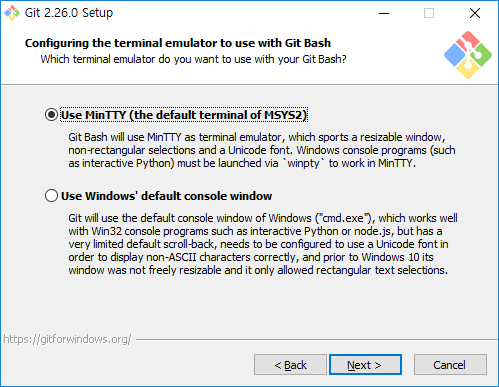
Use the native Windows Secure Channel library : Windows 인증서 저장소를 사용하여 서버 인증서의 유효성 검사



1. Checkout, Commit할때의 텍스트라인 엔딩을 선택하고 Next



1. Git Bash 터미널의 형식을 선택해줍니다.

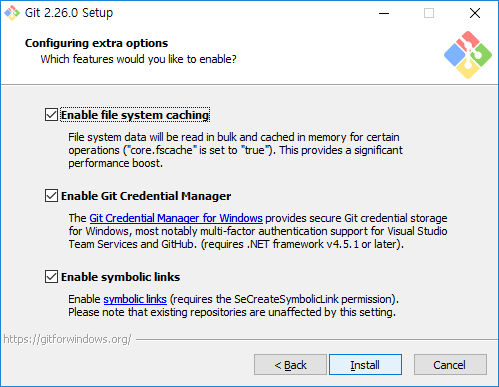


1. 옵션 선택합니다.

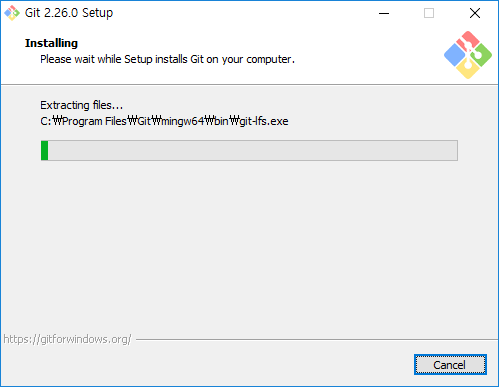
Enable file system caching : 성능향상을 위해 파일 시스템 데이터를 메모리에 캐시합니다.

Enable Git Credential Manager : Windows 용 보안 Git 자격증명 저장소를 사용하기 위해 Git Credential Manager 활성화합니다.

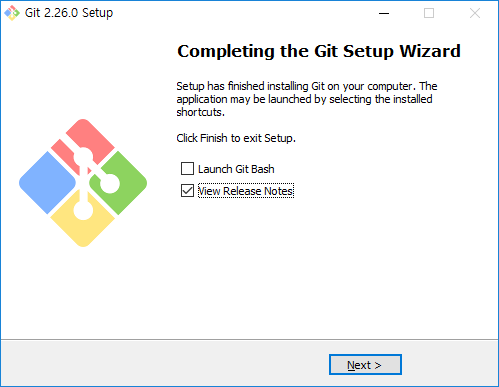
Enable symbolic links : symbolic links 활성화합니다.(기존 저장소는 영향을 받지 않습니다)



1. Install을 눌러 설치를 진행합니다.



1. 설치가 끝날때까지 조금만 기다려줍시다. 설치가 완료되었습니다.

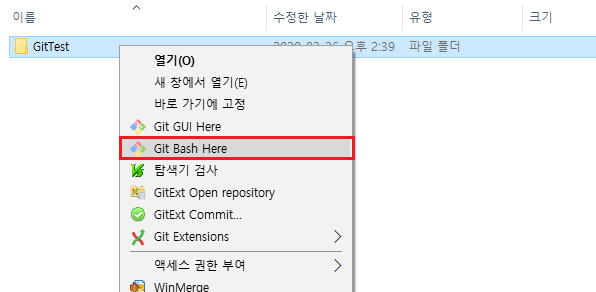


**Git 초기 설정**

Git을 처음 설치하면 Name, Email 초기 값 Null

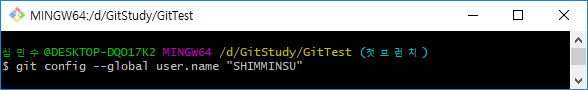
참고, user 설정이 되어 있지 않으면 GitHub에 있는 repository에 변경사항을 push 해도 commit count 집계도 안되고 해당 커밋의 작성자 프로필 아이콘도 ? 로 표시되기 때문에 name과 email 주소를 설정이 필요하다.

1. 원하는 소스폴더에서 Git Bash를 실행합니다.



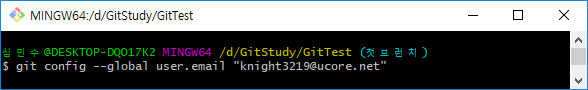
1. 실행한 Git Bash에 아래와 같이 명령어를 입력하여 Name을 설정합니다.

$git config --global user.name “[Name 입력]”



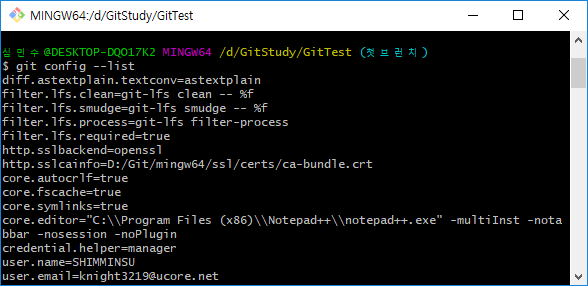
1. 아래와 같이 명령어를 입력하여 email을 설정합니다.

$git config --global user.email “[email 입력]”



1. 아래와 같이 명령어를 입력하여 List 內 Name, email 설정을 확인합니다.

$git config --list

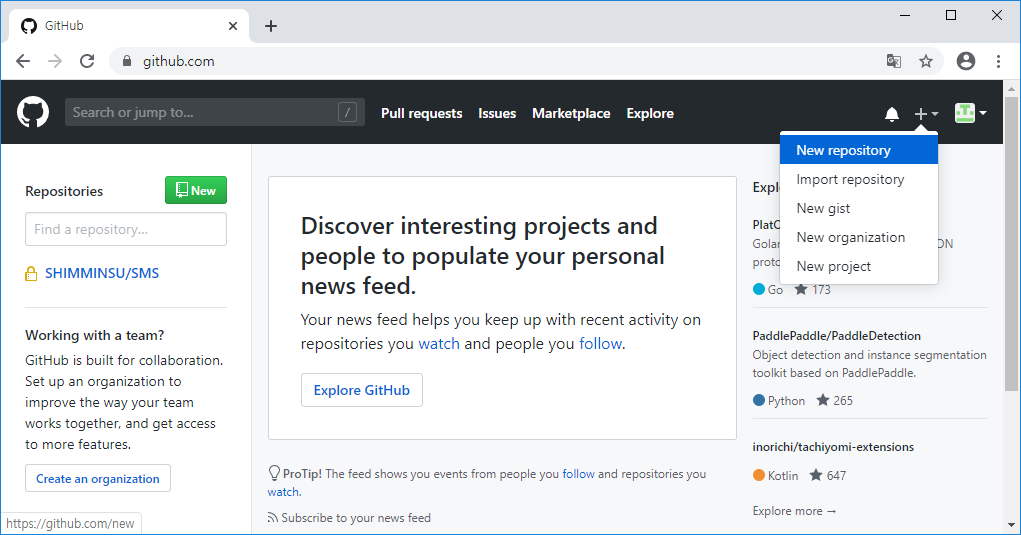


**GitHub 저장소(Repository) 생성 & 소스 올리기(Git Bash 활용)**

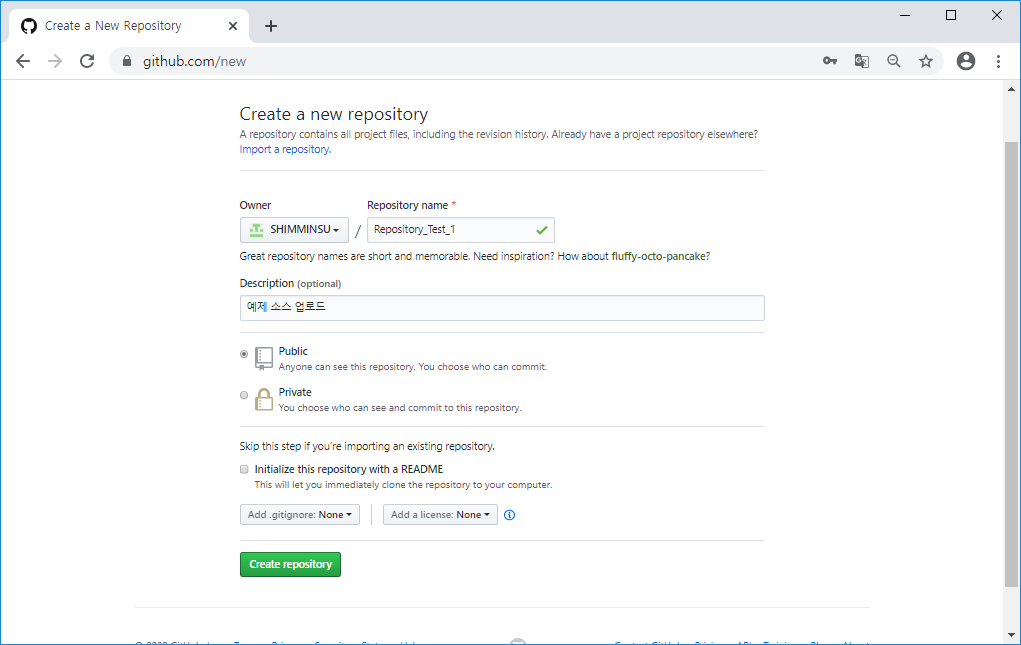
GitHub에서 원격 저장소를 만든 뒤 Git Bash를 활용하여 Push 하는 방법입니다.

**Git Repository 생성**

1. <https://github.com/> 링크된 깃허브 사이트에가서 로그인을 합니다. (회원이 아니라면 회원가입을 해주세요.) 그런 뒤 1시방향에 있는 +버튼을 눌러서 New repository버튼을 눌러줍니다.



1. repository 정보를 입력하고 Create repository버튼을 눌러줍니다.



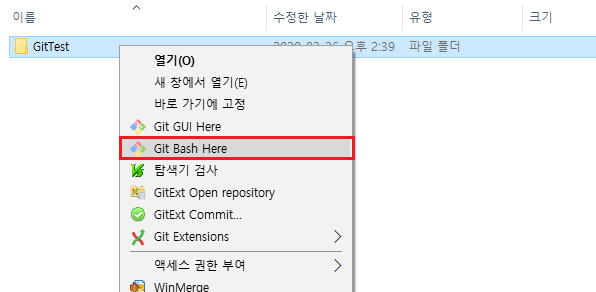
**공개, 비공개**

**설명**

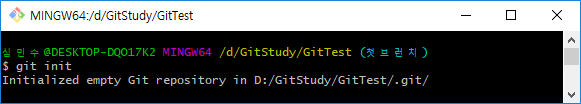
**주소**

**Git Bash에서 소스 Push**

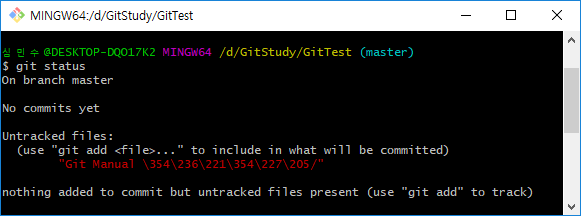
1. 소스폴더에서 Git Bash를 실행합니다.



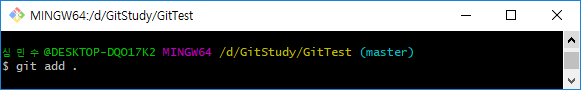
1. 깃을 init 시켜줍니다. (로컬저장소 만들기)



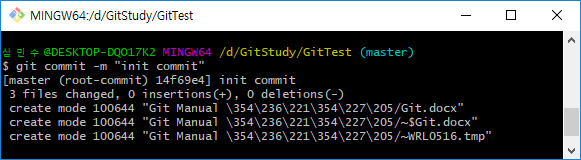
1. git status 올라가 파일이 있는지 현재 폴더에 있는 파일들을 확인합니다. 지금은 파일들이 빨간색으로 되어있지만 add작업을 하게되면 녹색으로 변합니다



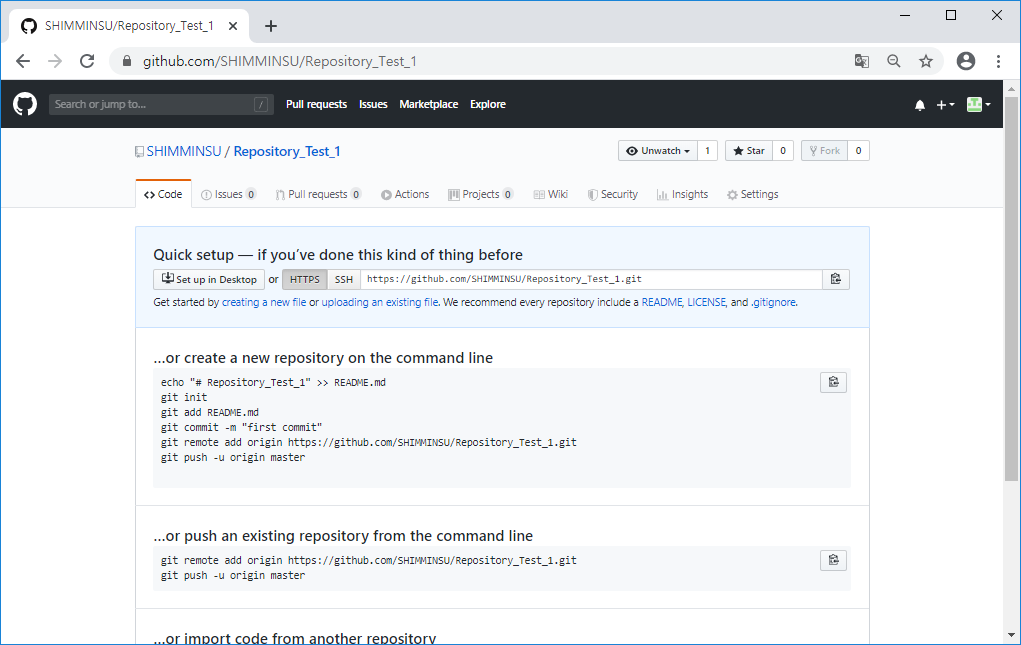
1. git add . 로컬저장소에 파일들을 올려줍니다. 모든 파일을 다 Push할것이기 때문에 add . 을 하였습니다. 특정 파일만 업로드를 하고싶으면 add [파일명 or 폴더명]이라고 해주시면 됩니다. add다음에 한 칸 띄우고 . 입니다.



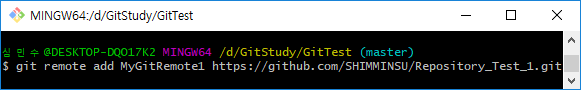
1. git commit -m "[Push 메시지명]" Push시 원하는 메시지를 등록하고 commit을 해줍니다.



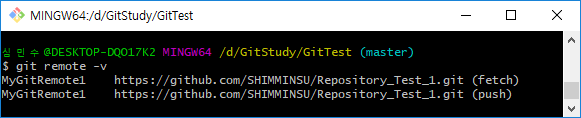
1. 자신이 올릴 깃 repository주소를 복사합니다. repository에서 오른쪽에 보이는 빨간박스안의 버튼을 누르면 복사할 수 있습니다.



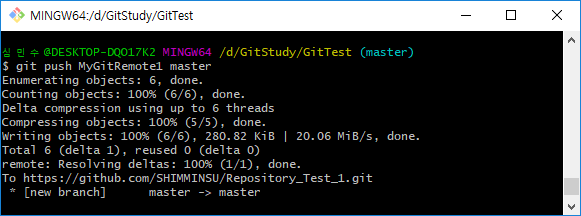
1. git remote add [명칭 지정] [repository 주소] 방금 복사해두었던 주소를 복사해줍니다. Ctrl+V는 안먹히니 마우스 오른쪽 클릭 -> paste을 사용합니다.



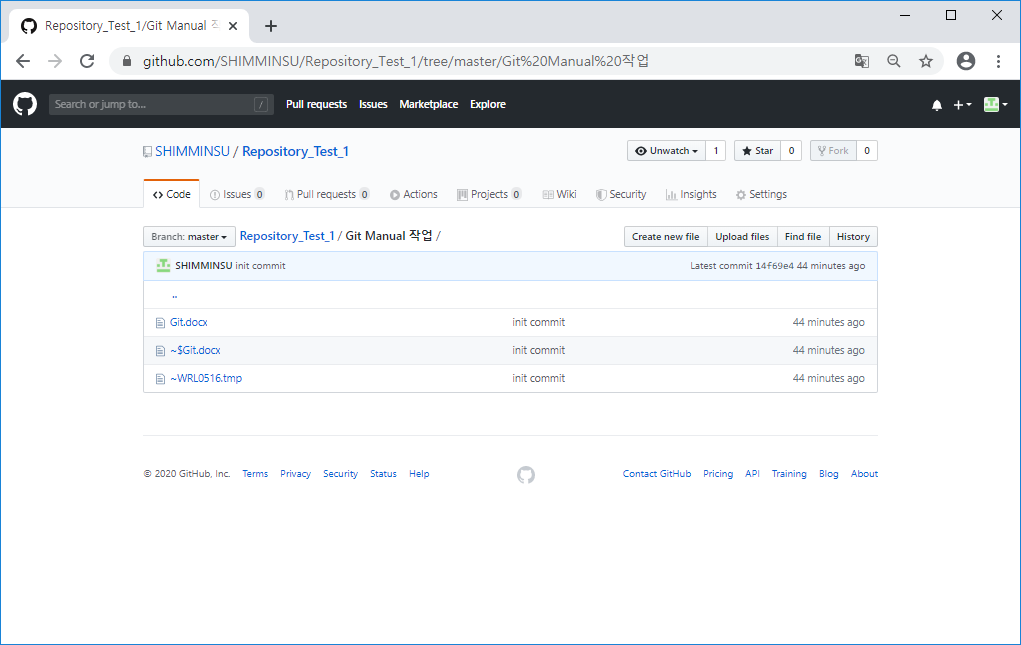
1. git remote –v 로컬 저장소의 원격 저장소 연결 정보 확인



1. git push [step.7 지정한 원격명칭] [브런치] 아까 commit 해두었던 파일들을 원격저장소에 올립니다.

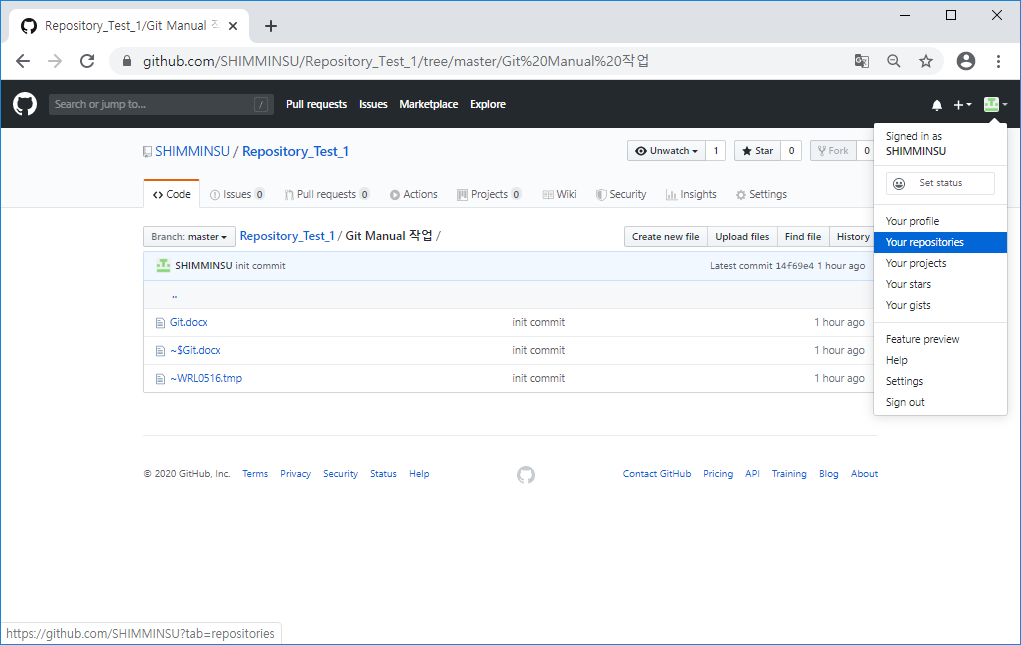


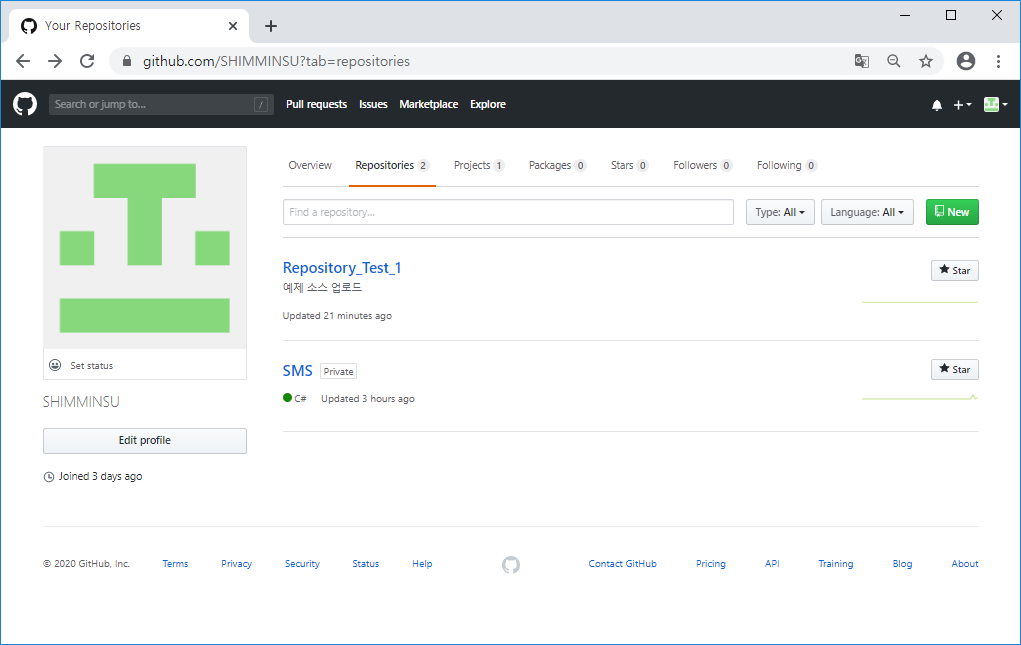
1. 파일들이 잘 업로드 되었는지 GitHub Code에서 확인 합니다.



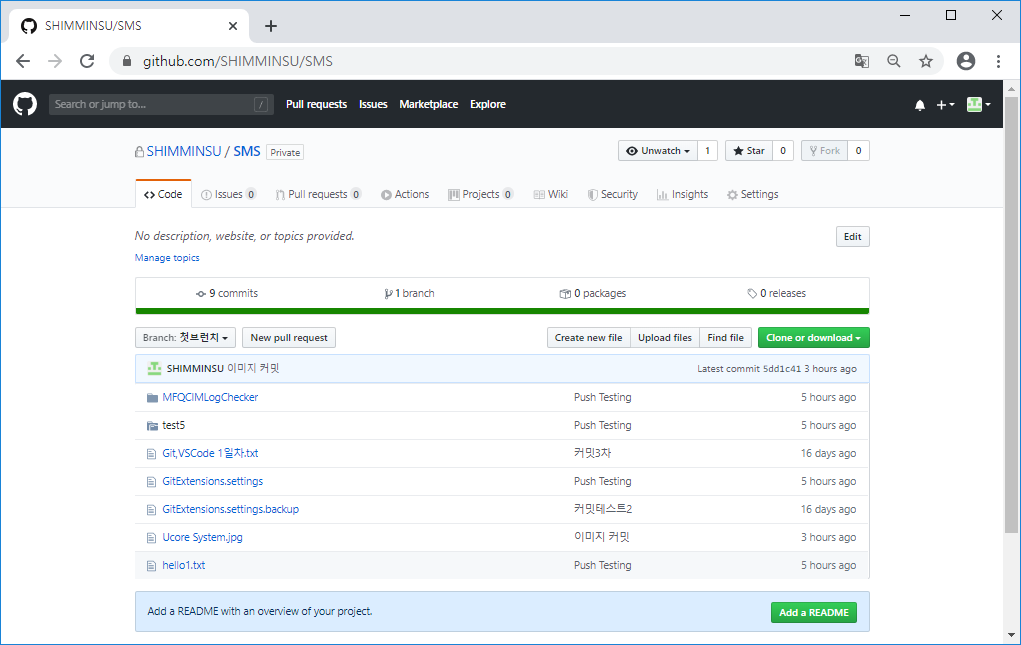
**GitHub 원격저장소 레파지토리(Repository) 삭제**

1. 깃허브 사이트에가서 로그인을 한 뒤 삭제하고 싶은 레파지토리로 들어갑니다.

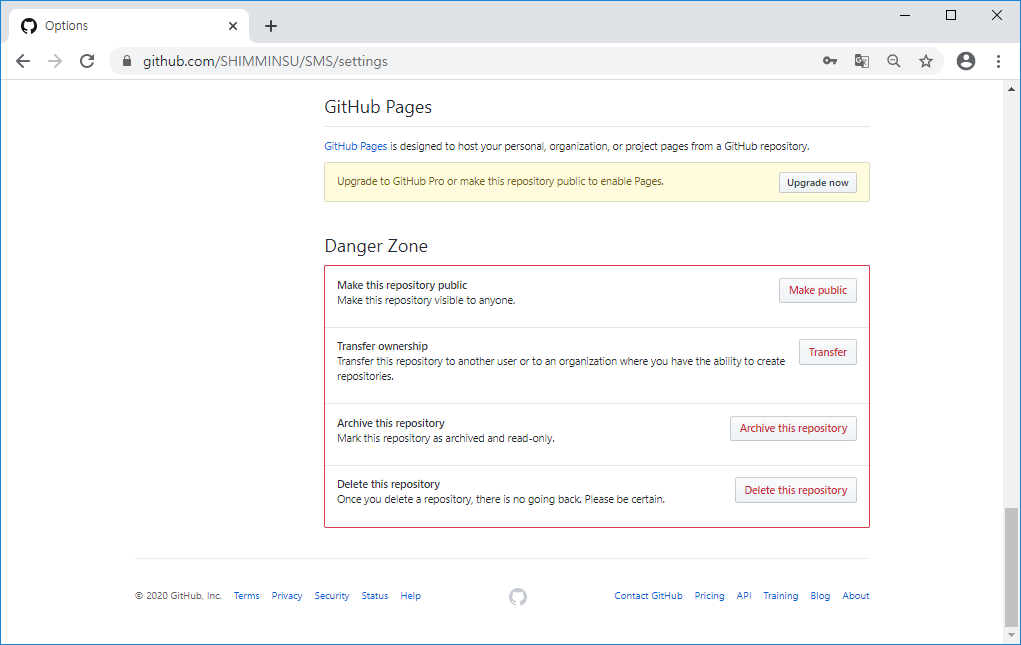




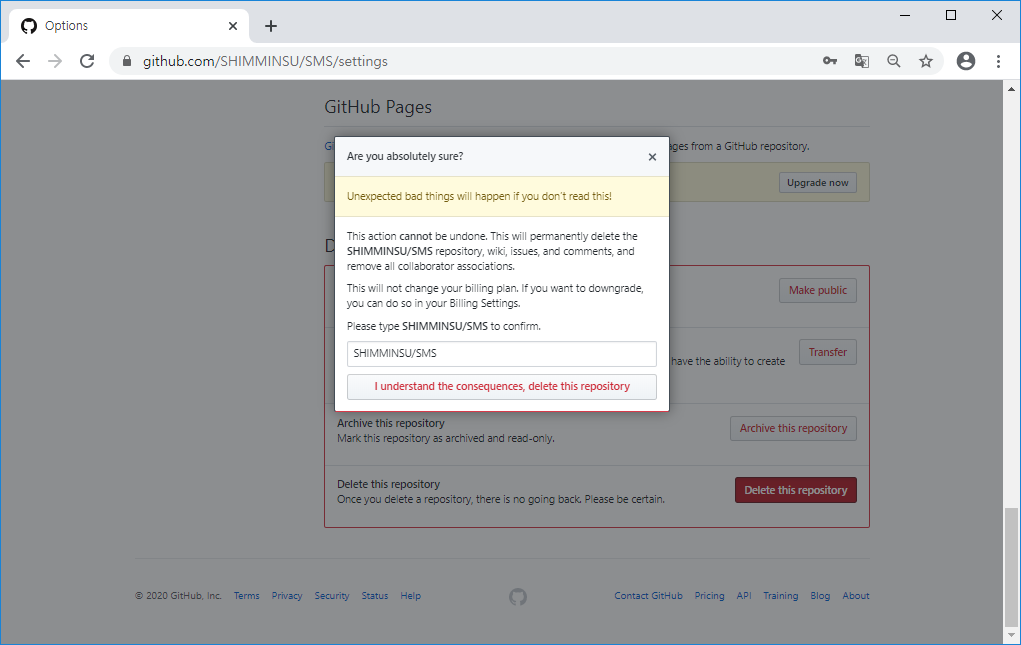
1. 삭제할 레파지토리에서 우측 상단의 Setting버튼을 클릭합니다.

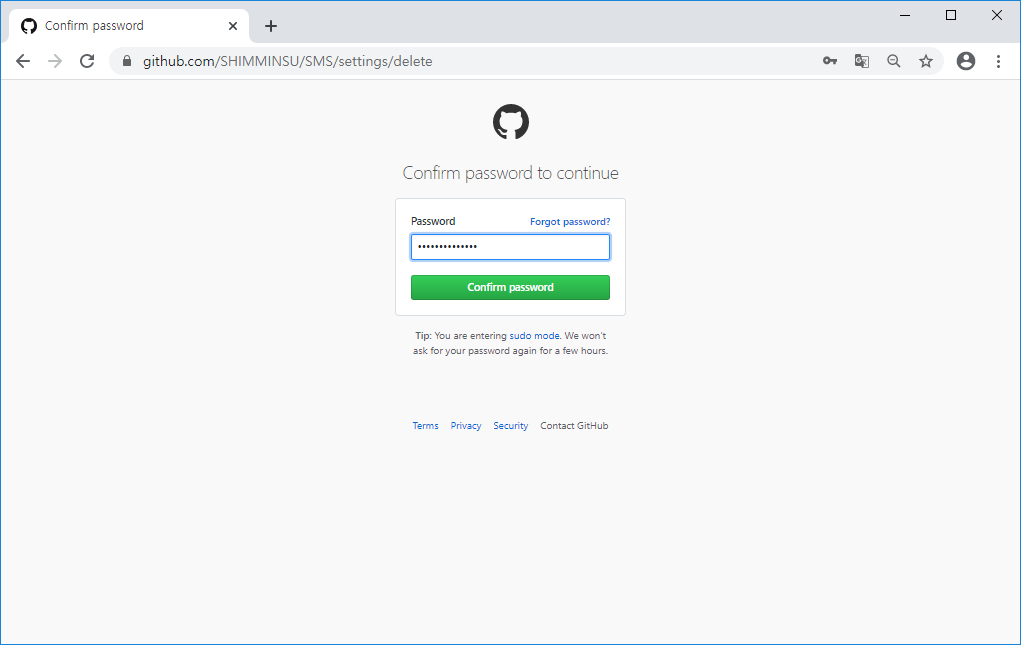


1. Setting에서 스크롤을 밑으로 끝까지 내리게되면 Danger Zone 內 가장 밑에있는 Delete this repository버튼을 클릭합니다.



1. 제거할 레파지토리명과 계정 패스워드를 차례로 입력해주면 레파지토리 삭제가 완료됩니다.





**<그림-21 GitHub Repository 삭제 전 계정 페스워드 입력>**

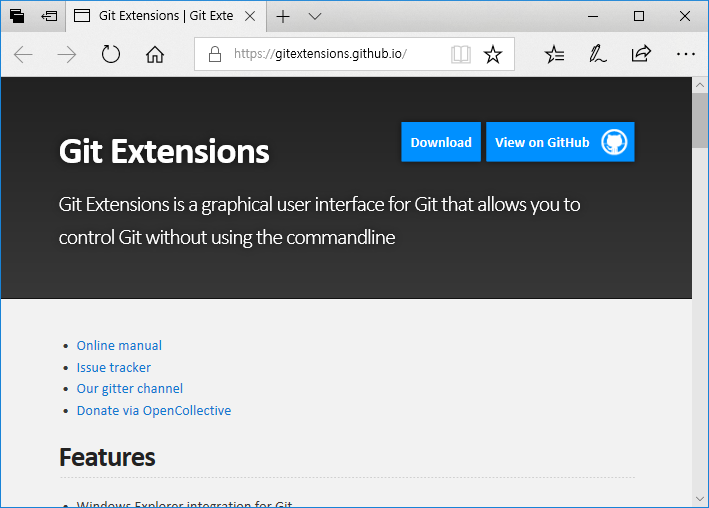
**Git Bash 명령어 정리**<https://git-scm.com/docs>

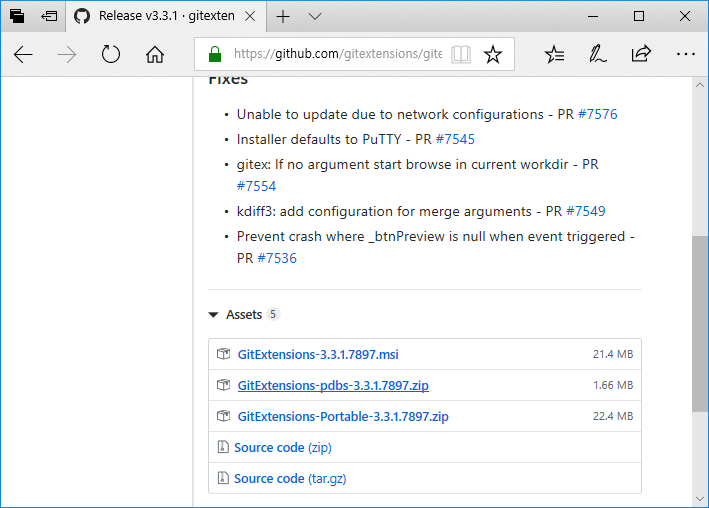
**Git Extensions**

Git Extensions와 깃을 연동하면 작업이 매우 편리해집니다. Git Extensions에서 몇 번의 클릭만으로 깃 레파지토리에 Push를 할수도 있고 Pull을 할 수도 있습니다.

**GIT Extensions 설치하기**

1. <https://gitextensions.github.io/> 링크 페이지에 들어가서 자신의 OS에 맞는 설치버전을 다운받는다.





1. 다운받은 zip 압축파일의 압출을 풀면 설치는 완료되고, 실행파일 실행 시 언어 선택 영문 지원.

**GIT Extensions 에서 소스 Push**

1. 설치한 GitExtensions 실행

