

GitHub

GitHub는 소프트웨어 개발 프로젝트를 위한 소스코드 관리 서비스입니다. 소스코드를 열람하고 간단한 버그 관리, SNS 기능까지 갖추고 있어 개발자에게 없어서는 안될 서비스입니다.

1. 코드 저장소
2. 소스코드 공유
3. 협업하는 공간

깃(Git)은 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템입니다.(여러 명의 개발자가 버전 관리를 하며 쉽게 협업할 수 있는 틀!)

분산 : 여러 개발 환경(즉 여러 명의 개발자)에서 한 프로젝트를 협업하며 개발할 수 있음

버전 : 개발 단계에 따라 한 프로젝트의 여러 버전을 나누고 관리할 수 있음

Repository: 코드나 문서를 비롯한 리소스를 저장하는 곳을 말하며, 프로젝트 단위로 만듭니다. 그냥 리포(repo)라고 줄여쓰기도 합니다.

원격 저장소(Remote repository) : 깃허브 같은 호스팅 서비스 서버에 올라가 있는 저장소

로컬 저장소(Local repository) : 개인 컴퓨터에 있는 저장소

Fork: 다른 사람의 원격 저장소를 그대로 복사해 내 계정의 원격 저장소로 만드는 것

Pull Request: 내 저장소의 변경 내용을 다른 사람의 저장소에 반영하도록 요청하는 것. 풀 리퀘스트를 보내면 해당 저장소의 메인테이너(프로젝트를 관리하는 사람)가 내 작업을 반영할지 말지 결정합니다. 풀 리퀘, PR이라고 줄여 말합니다.

Issue: 프로젝트의 버그 리포트, 기능 제안, 질문 등을 말하며, 깃허브 저장소에서 Issues 탭에 들어가면 다양한 토론을 볼 수 있습니다.

github.com
Sign Up

로그인후
Repository -- > New
Repository name-->프로젝트 이름
create repository

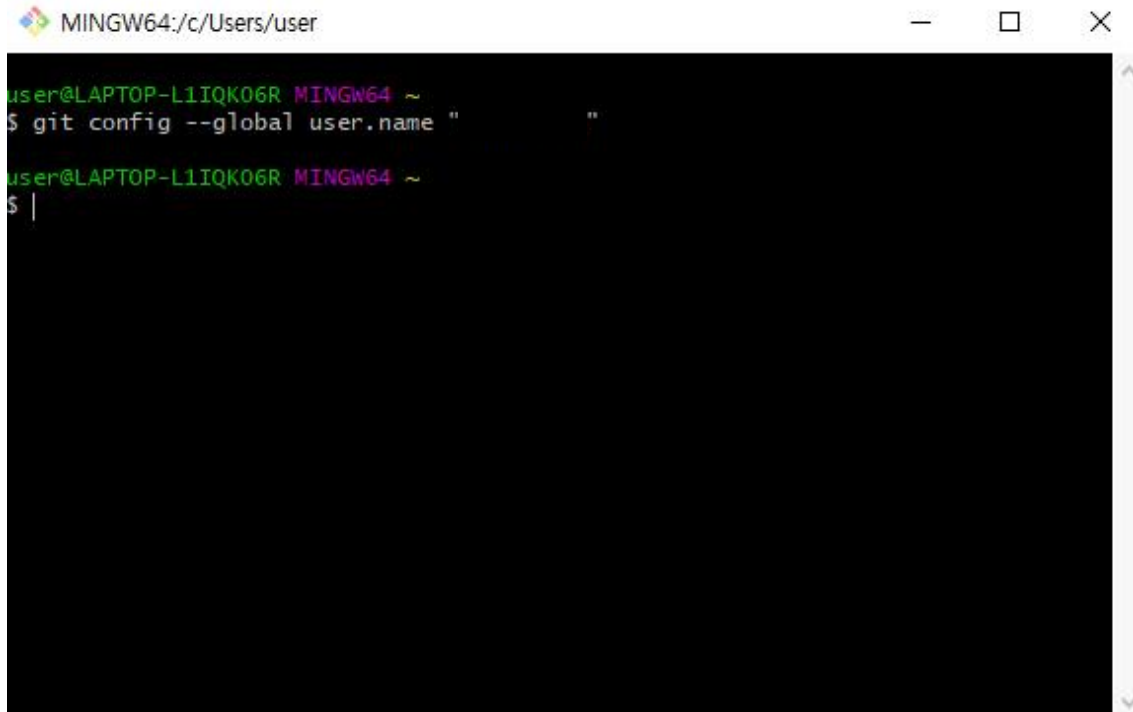
git 소스코드를 내 컴퓨터에서 인터넷으로 올려주는 역할
git다운로드

git-scm.com
download-->운영체제에 맞게 다운로드
윈도 메뉴 클릭후 git 설치 완료후 Git bash 열기

git bash에서 환경설정하기

Step 1 : 유저이름 설정

git config --global user.name "your_name"

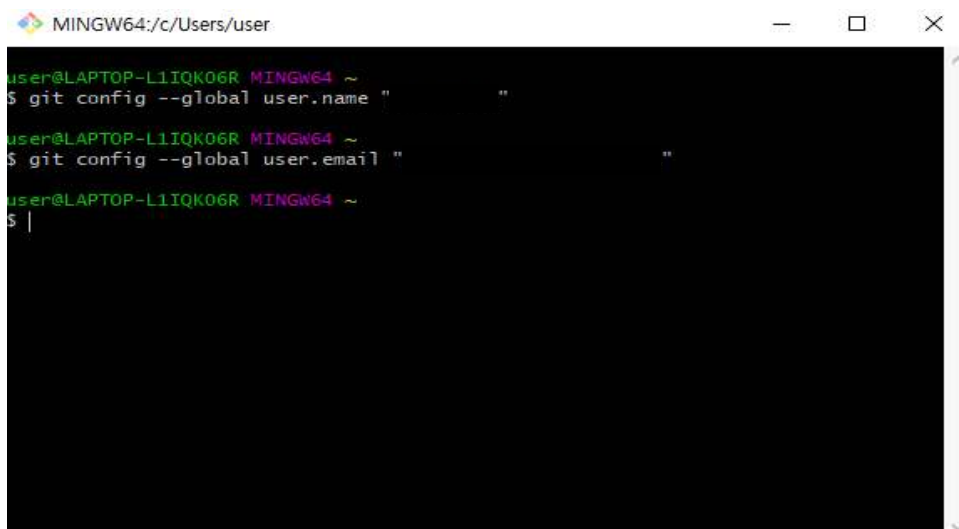


A screenshot of a terminal window titled "MINGW64:/c/Users/user". The prompt is "user@LAPTOP-L1IQK06R MINGW64 ~". The command "git config --global user.name " " is entered, followed by a second line " " to complete the command. The prompt returns to "user@LAPTOP-L1IQK06R MINGW64 ~".

Step 2 : 유저 이메일 설정하기

git config --global user.email "your_email"

Github가입시 사용한 이메일을 써주세요!



A screenshot of a terminal window titled "MINGW64:/c/Users/user". The prompt is "user@LAPTOP-L1IQK06R MINGW64 ~". The command "git config --global user.name " " is entered, followed by a second line " " to complete the command. The prompt returns to "user@LAPTOP-L1IQK06R MINGW64 ~". The command "git config --global user.email " " is entered, followed by a second line " " to complete the command. The prompt returns to "user@LAPTOP-L1IQK06R MINGW64 ~".

Step 3 : 정보 확인하기

git config --list

```
MINGW64:/c/Users/user
user@LAPTOP-L1IQK06R MINGW64 ~
$ git config --global user.name "Bitna Kim"

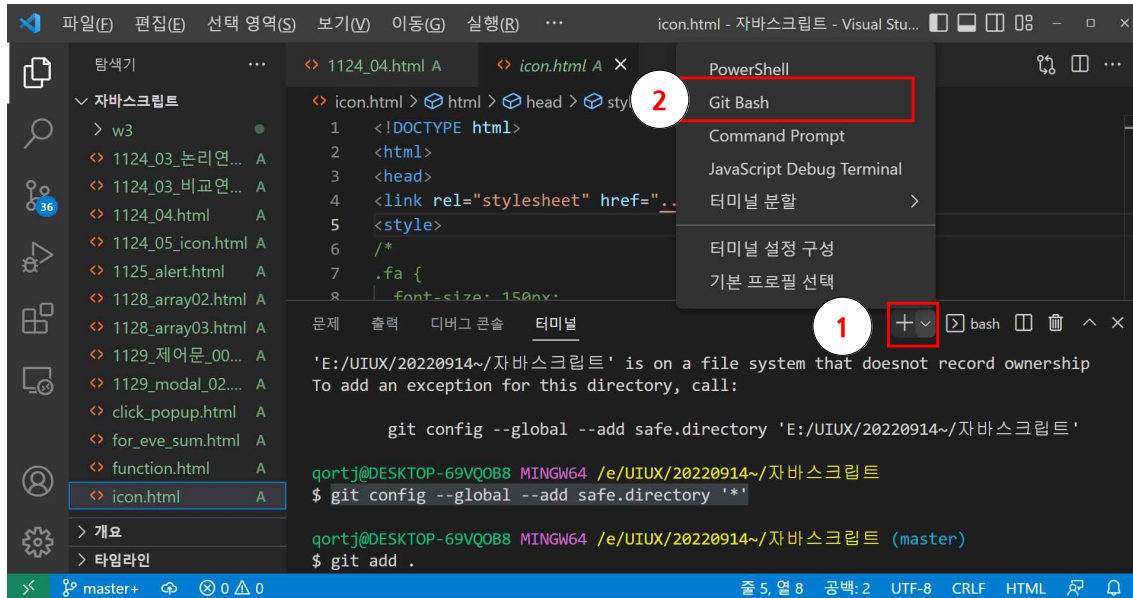
user@LAPTOP-L1IQK06R MINGW64 ~
$ git config --global user.email "tallshe108@hanmail.net"

user@LAPTOP-L1IQK06R MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager
user.email=tallshe108@hanmail.net
user.name=Bitna Kim
```

name과 email 확인

Github에 처음 코드 업로드하기

visual studio code-->Terminal(터미널) 메뉴-->새터미널



1. 초기화

git init-->맨처음에 프로젝트를 올릴때는 git init을 해줘야 합니다.

2. 추가할 파일 더하기

git add .-->.(점)은 모든 파일이라는 뜻, 선택적으로 올리고 싶으면 add 뒤에 파일 이름 붙여주면 됩니다.

3. 상태확인(선택사항)

git status-->상태를 알려주는 명령어

4. 히스토리 만들기

git commit -m "first commit"-->히스토리

-m은 메시지의 준말로 뒤에 ""안에 주고싶은 히스토리 이름을 주면 됩니다.

5. Github repository랑 내 로컬 프로젝트랑 연결하기

git remote add origin https://github.com/*****

이 명령어는 github에서 복사해서 붙여와도됩니다.

Repository로 내 소스코드를 보낸다 라는 뜻 연결고리

6. 잘 연결됐는지 확인(선택사항)

git remote -v

내가 연결한 주소값이 잘 뜨면 성공

7. Github로 올리기

git push origin master

Github에 계속 업데이트 하는법

1. 추가할 파일 더하기

`git add .`

`git status`

2. 히스토리 만들기

`git commit -m "second commit"`

3. Github로 올리기

`git push origin master`

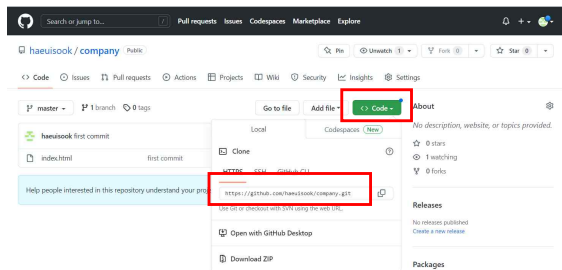
내 컴퓨터에 소스코드를 업데이트를 하고 싶으면 이 **세개의 스텝만** 계속 반복하면 됩니다.

Github로 팀프로젝트 하는법

Github에서 소스코드 다운로드

1. 명령프롬프트 실행

2. `git clone` 주소 폴더이름



- 주소는 깃허브에서 들고와야합니다
- 폴더이름은 선택사항입니다. 폴더이름을 지정하면 폴더가 새로 생성이 되면서 그 안에 코드들이 다운로드가 되고, 폴더이름을 생략하면 깃허브 프로젝트 이름으로 폴더가 자동으로 생기고 그안에 코드들이 다운로드 됩니다.

3. copy가 완료되면

4. `cd` 폴더이름입력

5. `code .` --> 입력하면 visual studio code 실행됩니다.

6. `git add .`(`git clone`할때 이미 세팅 정보도 같이 와서 `init`, `remote`하지 않아도 됨)

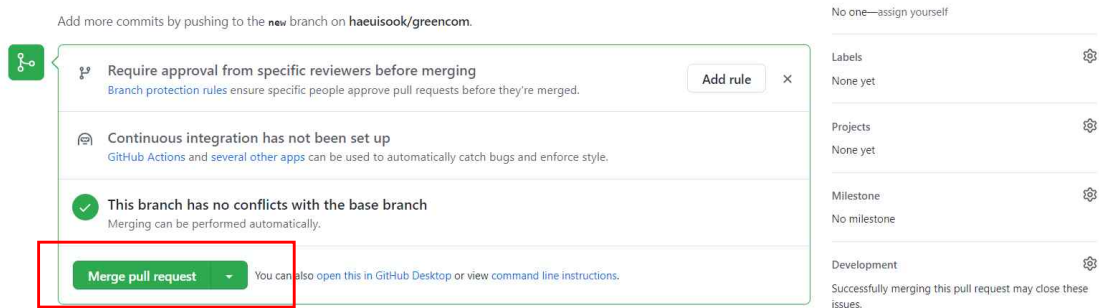
7. `git commit -m "new_man first commit"`

8. Github에서 내 브랜치(branch)만들기

`git checkout -b` 브랜치이름

9. `git push origin` 브랜치이름

10. 개발리더가 Merge Pull requests 클릭 confirm Merge



11. 개발리더 동기화 작업

git add .

git commit -m "second commit"

git pull origin master(마스터 브랜치로부터 새로운 코드를 받아온다)

git push origin master(작업 후 다시 업로드)