

# 깃허브(기본 기능)

깃허브는 소프트웨어 개발에서 코드를 관리하고 기록하여 체계적인 개발이 가능하도록 도와주는 공개 소프트웨어입니다.



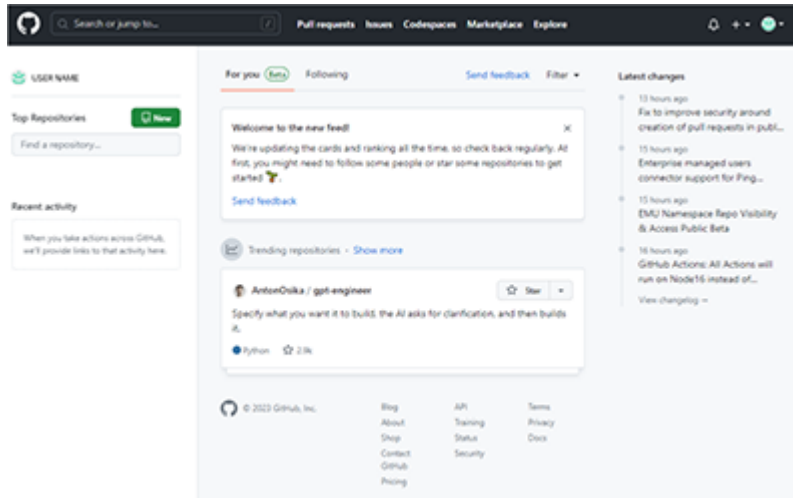
<https://gitforwindows.org/> 파일 다운로드 후 설치



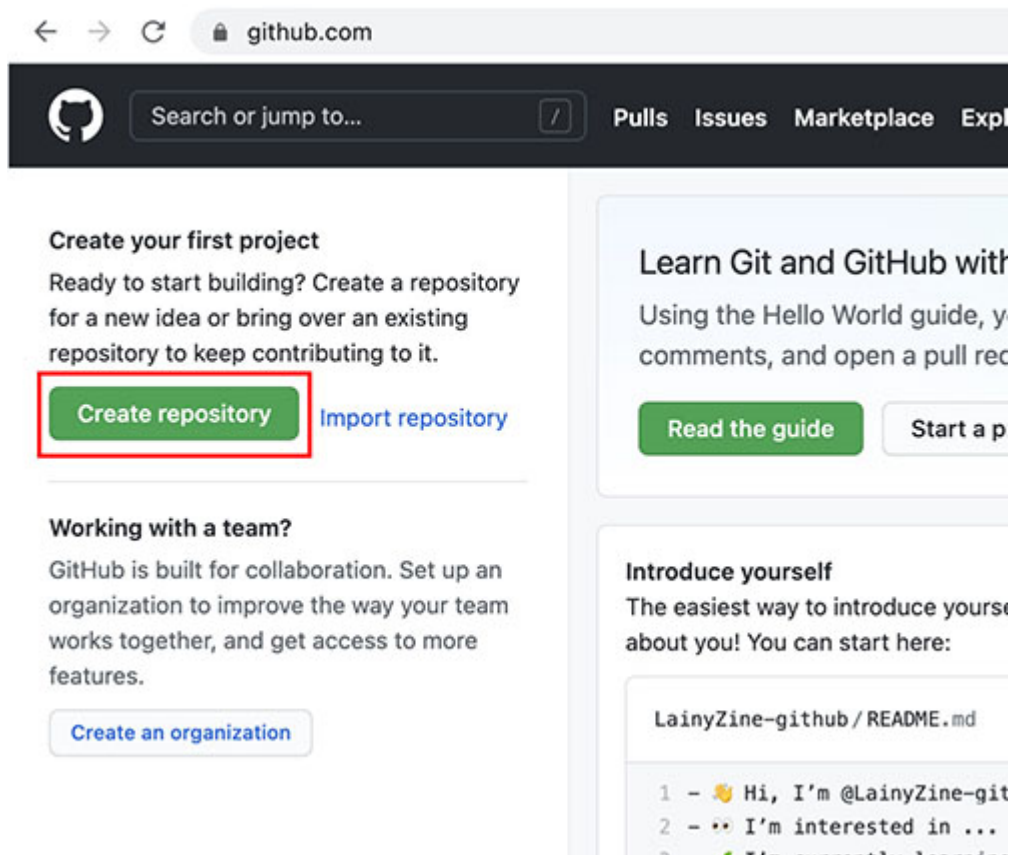
## github에서 계정 생성



<https://github.com/> 사이트에서 계정 생성(이메일 인증)



## 저장소(repository) 생성




“Create repository” 버튼을 클릭하여 새로운 저장소 생성

## Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \*

 USER NAME

Repository name \*

저장소 이름 입력

Great repository names are short and memorable. Need inspiration? How about [congenial-computing-machine?](#)

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:



Add a README file

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

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

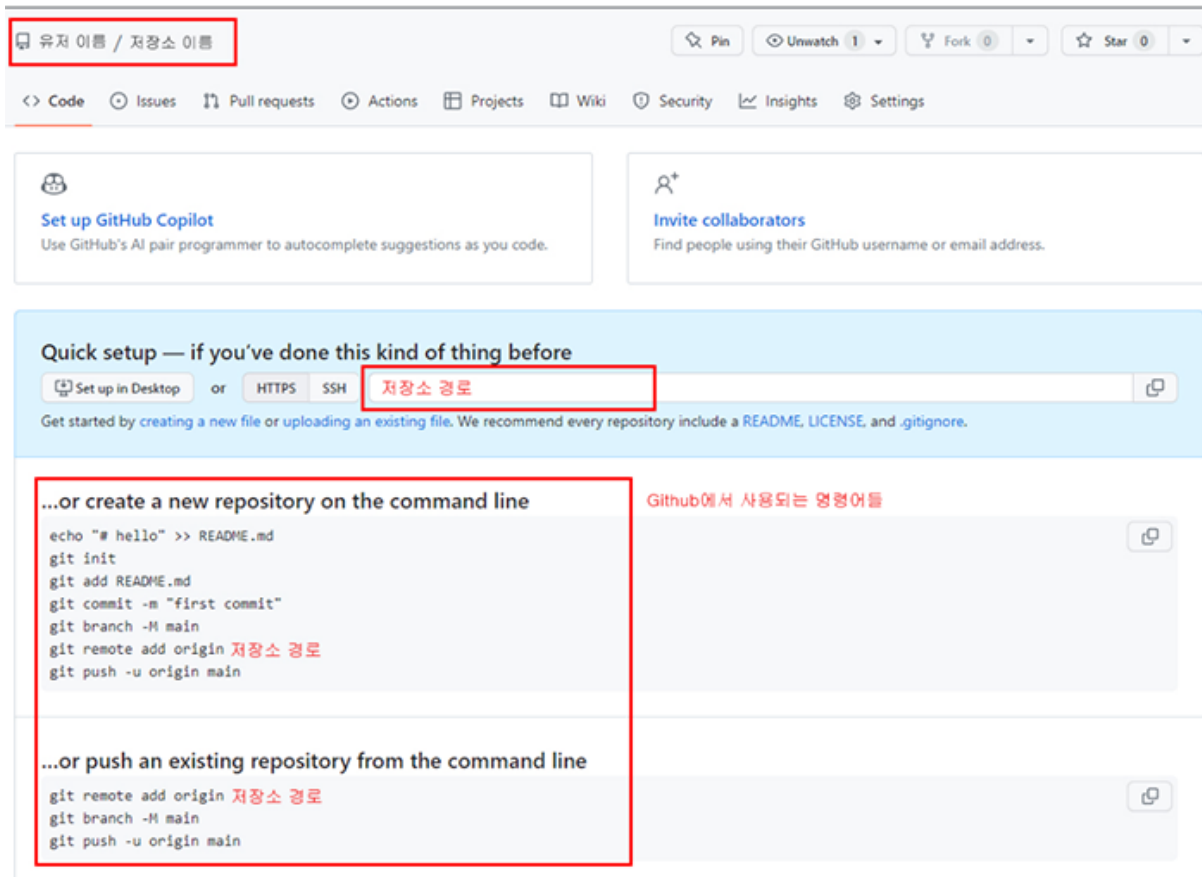
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

Create repository


저장소 이름을 입력하고 Create repository 버튼을 클릭



저장소 생성 완료

## git bash 또는 vscode를 활용하여 로컬 위치 설정

### git bash

 MINGW64:/c/ex

```
Administrator@User-2023CYLRAV MINGW64 ~  
$ cd c:  
  
Administrator@User-2023CYLRAV MINGW64 /c  
$ mkdir ex  
  
Administrator@User-2023CYLRAV MINGW64 /c  
$ cd ex  
  
Administrator@User-2023CYLRAV MINGW64 /c/ex  
$ |
```

예제는 c드라이브에 ex폴더를 만들고 ex폴더로 이동하였음.



cd = 디렉토리 이동 명령어



mkdir = 새로운 디렉토리를 만드는 명령어

## | vscode



"파일 > 폴더 열기"로 경로를 설정하고 터미널 메뉴에서 새 터미널을 꺼내주면 됨.

## 현재 디렉터리를 Git 저장소로 변환

- 현재 폴더를 git저장소로 설정합니다.

```
git init
```

```
PS E:\ex> cd hello
PS E:\ex\hello> git init
Reinitialized existing Git repository in E:/ex/hello/.git/
```

예제는 ex/hello폴더를 git저장소로 변환하였다.

### "유저 이름"과 "가입할 때 사용한 메일 주소"를 설정

```
git config --global user.name "유저 이름"
```

```
git config --global user.email "가입할 때 사용한 메일 주소"
```



git config = 설정 내용을 확인하고 변경할 수 있는 명령어

## 원격 저장소 추가 및 확인하기

- repository 제거와 추가

```
# 새 리포지토리 추가
git remote add origin https://github.com/<user-name>/<repository-name>.git

# 기존 리포지토리 제거
git remote remove origin
```

- 현재 프로젝트에 등록된 리모트 저장소를 확인한다.

```
git remote
```

저장소를 clone 하면 origin이라는 리모트 저장소가 자동으로 등록되기 때문에 origin이라는 이름을 볼 수 있다.

- 원격 저장소 경로 확인

```
git remote -v
```

리모트 저장소가 여러 개 있다면 등록된 전부를 보여준다.

```
PS E:\ex\hello> git remote add origin https://github.com/user/repository.git
PS E:\ex\hello> git remote -v
origin https://github.com/user/repository.git (fetch)
origin https://github.com/user/repository.git (push)
```



## 파일을 원격 저장소에 추가

- 원격 저장소에 추가(실제 추가 되는게 아니라 깃 저장소의 스냅샷에 포함 됨)

```
// . = 폴더안의 모든파일 (파일명을 입력하여 원하는 파일만 선택할 수 있음)  
git add .
```

```
//파일이 추가되어 있는지 확인  
git status
```

| git status = 파일이 추가되어 있는지 확인 합니다.

```
PS E:\ex\hello> git add hello.html  
PS E:\ex\hello> git status  
On branch main  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   hello.html
```

예제는 hello.html파일을 생성하고 추가하였음.

# 커밋(파일이나 폴더의 추가 또는 변경을 저장소에 기록하는작업)

```
git commit -m "기록할 메시지"
```

```
//기본 브랜치 명을 main으로 설정(설치하고 처음에 한번 작성해 주어야 함)  
git branch -M main
```

```
PS E:\ex\hello> git commit -m 'first commit'  
[main (root-commit) 4faff15] first commit  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 hello.html
```

예제는 커밋 메시지를 "first commit" 이라 작성하였다.

## 커밋 메시지 변경

- `git commit --amend` 명령을 사용하여 가장 최근의 커밋 메시지를 변경할 수 있습니다.

```
git commit -m "변경할 메시지" --amend
```

## 로컬 저장소에 있는 파일을 원격 저장소에 반영

```
git push origin main
```

```
PS E:\ex\hello> git push -u origin main
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), 407 bytes | 407.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/user-name/repository.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

원격 저장소로 push

---

**위 과정은 초기에 한번만 하면 됩니다.**

**이후에 수정된 문서를 깃 허브로 push 할 경우**

```
# add, commit, push만 입력하면 됩니다.
git add .

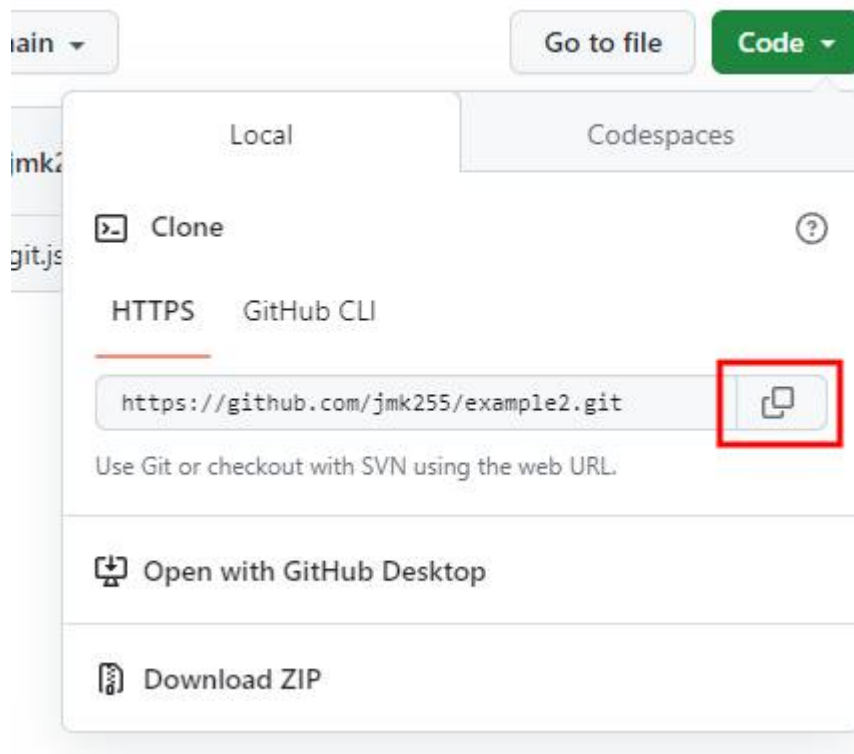
git commit -m "기록할 메시지"

git push origin main
```

---

**github저장소에 있는 내용을 가져오기**

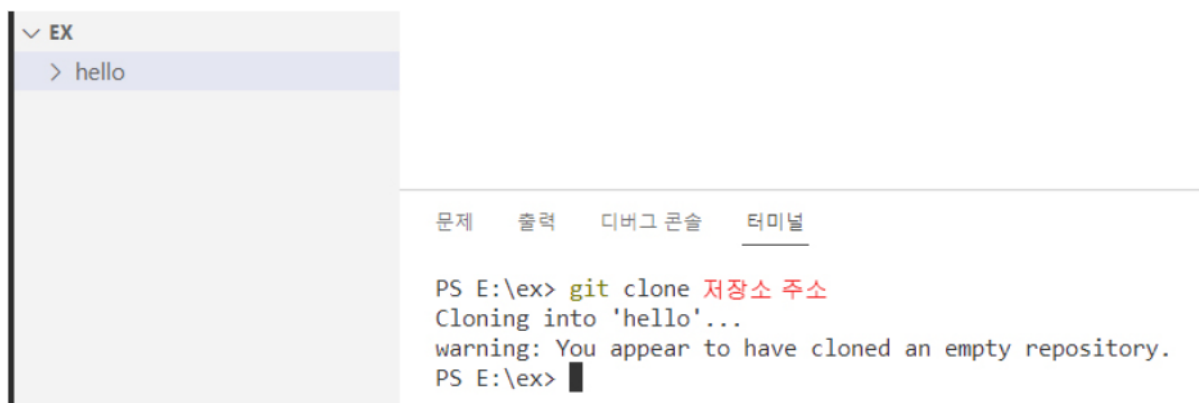
**1. 복제할 저장소 경로를 복사함**



```
git clone "저장소 주소" new_branch(저장 폴더 이름)
```

```
cd new_branch
```

```
code . //해당폴더로 vscode를 실행
```



예제는 hello 라는 이름의 저장소를 생성하고 로컬드라이브의 ex라는 폴더에 내용을 가져옴

## 이후 코드 작성 후 파일 업로드

"파일 생성 후 추가"

```
git add .
```

"커밋"

```
git commit -m "new_branch commit"
```

"푸시"

```
git push origin new_branch
```

## 리모트 저장소를 Pull 하거나 Fetch 하기

- 리모트 저장소에서 데이터 가져오기
- `git pull` 명령은 Clone 한 서버에서 데이터를 가져오고 그 데이터를 자동으로 현재 작업하는 코드와 `Merge` 시킨다.

```
git pull <remote>
```

"<remote> = 리모트 저장소 이름"

- 이 명령은 로컬에는 없지만, 리모트 저장소에는 있는 데이터를 모두 가져온다.
- `git fetch` 명령은 리모트 저장소의 데이터를 모두 로컬로 가져오지만 자동으로 `Merge` 하지 않는다.

```
git fetch origin
```