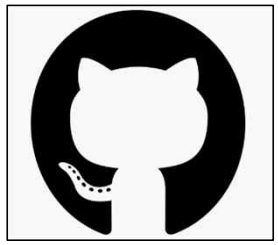
Github 사용법



1. Github 소개

1. GitHub 개요

GitHub는 Git 기반 분산 버전 관리 시스템을 사용하는 소프트웨어 개발 협업 플랫폼이다.
전 세계 개발자들이 소스 코드를 공유하고, 협업하며, 프로젝트를 관리하는 데 사용한다.
현재는 개인 개발자뿐 아니라 기업, 공공기관, 교육기관에서도 표준적인 개발 플랫폼으로 활용된다.

---

2. GitHub 운영 주체

GitHub는 **Microsoft(MS)**가 운영하는 플랫폼이다.

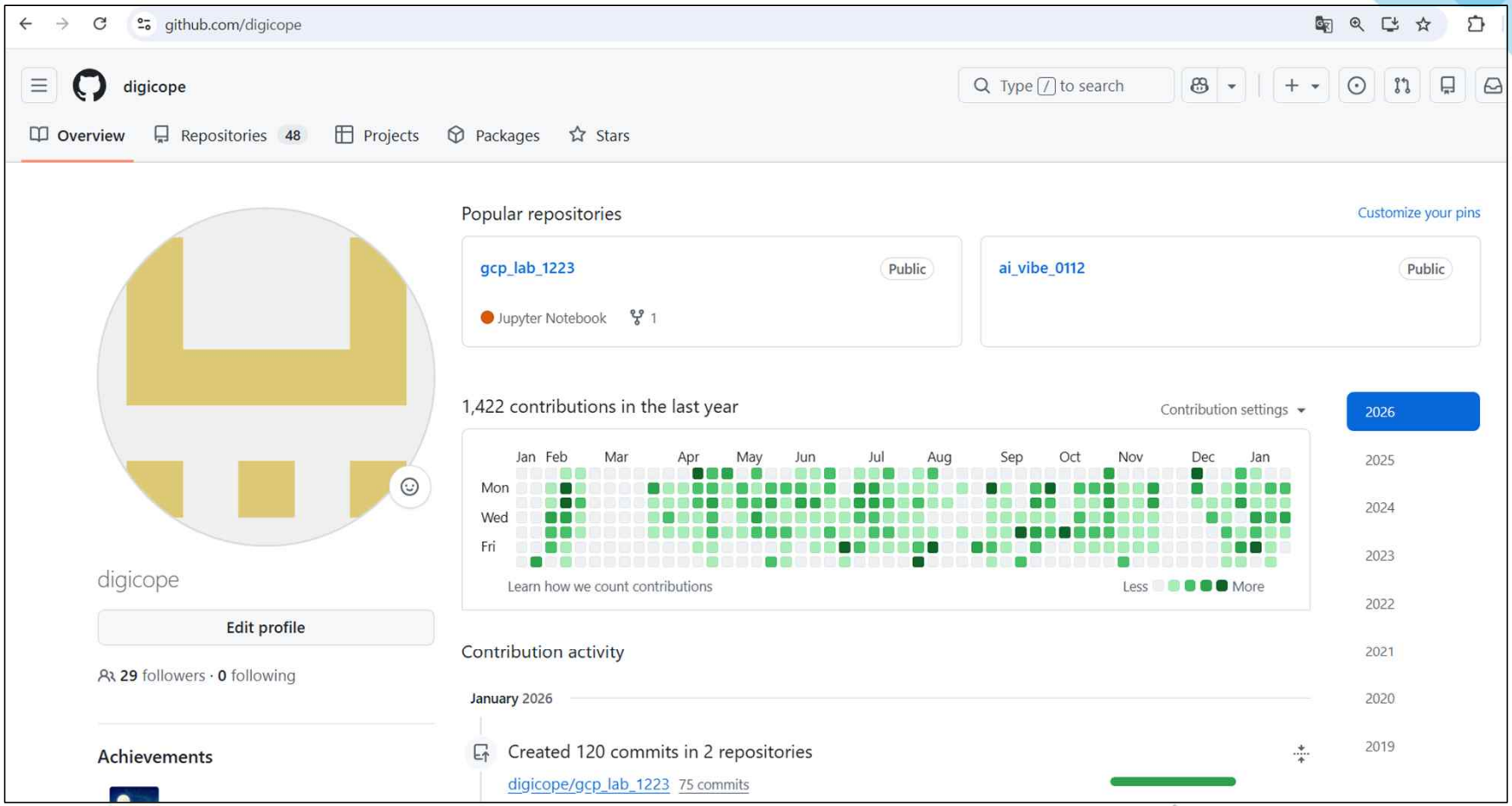
- GitHub는 2008년 설립된 독립 기업이었다.
- 2018년 Microsoft가 GitHub를 약 75억 달러에 인수하였다.
- 인수 이후에도 GitHub는 독립적인 브랜드와 운영 정책을 유지하고 있다.
- Microsoft 산하 조직이지만, 특정 언어나 플랫폼에 종속되지 않는 중립성을 유지한다.

Microsoft는 GitHub를 자사 클라우드(Azure), 개발 도구(Visual Studio, VS Code)와 연계하여 개발 생태계를 확장하고 있다.

3. GitHub의 핵심 역할

GitHub는 다음과 같은 역할을 수행한다.

- 소스 코드 버전 관리
- 협업 개발 환경 제공
- 오픈소스 프로젝트 허브
- 이슈 관리 및 프로젝트 관리
- CI/CD 및 자동화 연계
- 개발자 포트폴리오 플랫폼


github.com/digicope
digicope
Type / to search
Overview
Repositories 48
Projects
Packages
Stars
Popular repositories
Customize your pins
gcp_lab_1223
Public
Jupyter Notebook
1
ai_vibe_0112
Public
1,422 contributions in the last year
Contribution settings
2026
2025
2024
2023
2022
2021
2020
2019
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions
Less More
Contribution activity
January 2026
Created 120 commits in 2 repositories
digicope/gcp_lab_1223 75 commits
digicope
Edit profile
29 followers · 0 following
Achievements

4. GitHub 운영 정책 개요

4.1 오픈소스 중심 정책

GitHub는 오픈소스 생태계 활성화를 핵심 가치로 둔다.

- 공개(Public) 저장소는 무료로 무제한 생성 가능
- 오픈소스 프로젝트에 대한 접근과 기여를 장려
- 전 세계 개발자가 코드 열람 및 참여 가능

4.2 중립성 유지 정책

Microsoft 소유이지만 다음 정책을 유지한다.

- Linux, Python, Java, PHP 등 모든 언어를 동등하게 지원
- Azure 사용을 강제하지 않음
- AWS, GCP 프로젝트도 자유롭게 호스팅 가능

4.3 보안 및 신뢰 정책

- 2단계 인증(2FA) 지원
- 취약점 자동 탐지(Dependabot)
- 코드 서명 및 감사 로그 제공
- 기업용 보안 정책 및 접근 제어 기능 제공

5. GitHub 수익 구조

GitHub의 주요 수익원은 **유료 구독 서비스**이다.

5.1 개인 및 팀 요금제

- Free: 개인 사용자 및 공개 프로젝트 중심
- Pro: 개인 개발자용 유료 플랜
- Team: 소규모 팀 협업용
- Enterprise: 기업용 대규모 협업 및 보안 기능

유료 플랜에서는 다음 기능을 제공한다.

- 비공개 저장소 고급 권한 관리
- 고급 보안 및 감사 기능
- 조직 단위 관리 기능
- SLA 및 기술 지원

5.2 GitHub Enterprise

대기업 및 기관을 위한 수익 모델이다.

- 사내 개발 환경 통합
- SSO, LDAP, AD 연동
- 감사 로그 및 규정 준수 기능
- 대규모 조직 관리 기능

5.3 GitHub Copilot

AI 기반 코드 자동 완성 서비스이다.

- OpenAI 기술 기반
- 월 단위 또는 연 단위 구독 과금
- 개인, 팀, 기업 요금제로 제공

GitHub Copilot은 최근 GitHub의 핵심 성장 수익원 중 하나이다.

6. Microsoft와 GitHub의 전략적 관계

Microsoft는 GitHub를 다음과 같이 활용한다.

- Azure 클라우드 사용 확산
- 개발자 생태계 확보
- AI 개발 도구 확장
- DevOps 시장 경쟁력 강화

GitHub는 Microsoft의 클라우드 및 AI 전략에서 핵심 플랫폼 역할을 수행한다.

---

7. GitHub의 주요 사용자층

- 개인 개발자
- 스타트업
- 대기업 개발 조직
- 오픈소스 커뮤니티
- 교육기관 및 학생
- 공공기관 및 연구기관

8. GitHub의 현재 위상

GitHub는 사실상 **전 세계 표준 개발 협업 플랫폼**이다.

- 수억 개의 저장소 보유
- 수천만 명의 개발자 사용자
- 대부분의 주요 오픈소스 프로젝트가 GitHub에서 관리됨

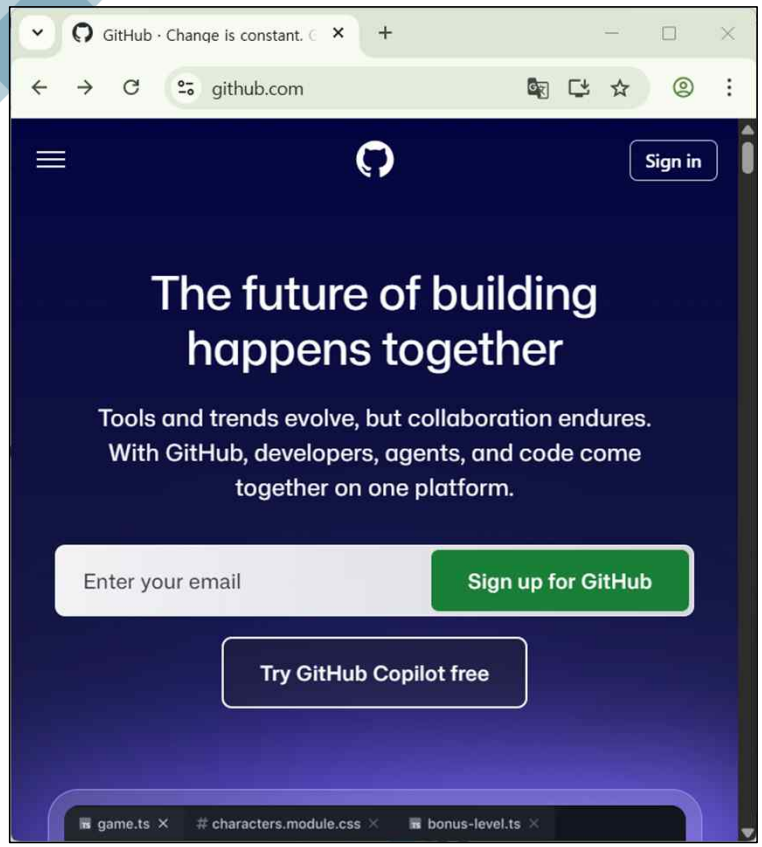
---

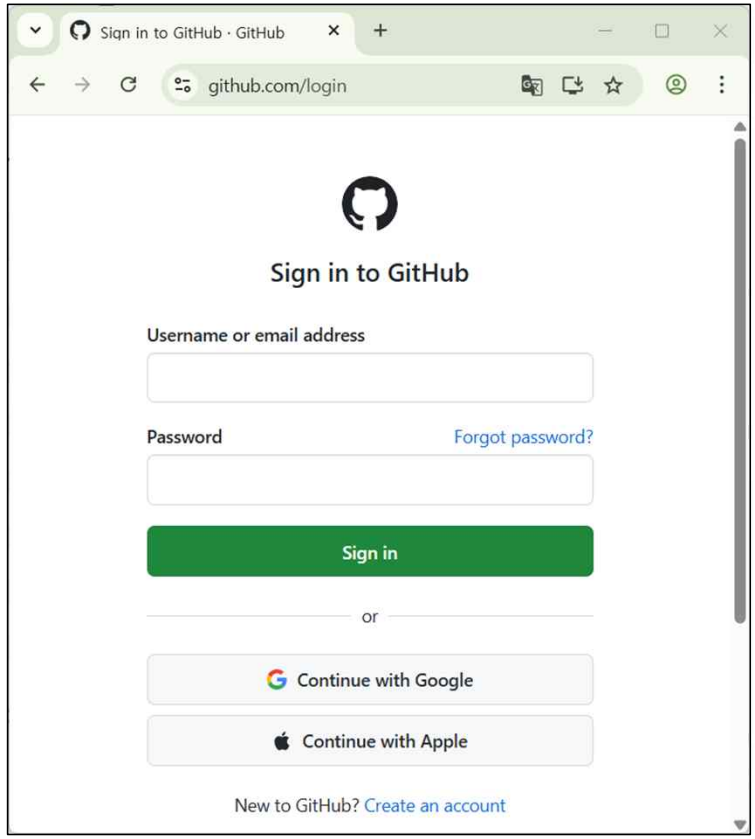
9. 정리

GitHub는 Microsoft가 운영하는 글로벌 개발 협업 플랫폼이다.
오픈소스 중립성을 유지하면서도, 기업용 서비스와 AI 기반 기능을 통해 수익을 창출한다.
현대 소프트웨어 개발 환경에서 필수적인 인프라 역할을 수행한다.

2. Github 계정 가입 및 사용법

https://github.com/





1. GitHub란 무엇인가

GitHub는 Git이라는 버전 관리 시스템을 기반으로 한 코드 저장 및 협업 플랫폼이다.
소스 코드의 변경 이력을 관리하고, 여러 사람이 함께 프로젝트를 개발할 수 있도록 지원한다.
개발자뿐 아니라 문서 작성, 데이터 분석, 포트폴리오 관리 용도로도 널리 사용된다.

2. GitHub 회원가입 방법

2.1 GitHub 접속

1. 웹 브라우저에서 https://github.com ↗ 에 접속한다.
2. 화면 오른쪽 상단의 Sign up 버튼을 클릭한다.

2.2 계정 정보 입력

다음 정보를 순서대로 입력한다.

- Email address : 사용할 이메일 주소
- Password : 비밀번호 (영문, 숫자, 특수문자 조합 권장)
- Username : GitHub에서 사용할 아이디

입력 후 Continue 버튼을 눌러 다음 단계로 진행한다.

2.3 이메일 인증

- 입력한 이메일 주소로 인증 메일이 전송된다.
- 메일에 포함된 인증 코드를 입력하여 계정을 활성화한다.

2.4 요금제 선택

- Free 플랜을 선택하면 된다.
- 무료 플랜에서도 개인 프로젝트와 공개 저장소 사용이 가능하다.

4. GitHub 저장소(Repository) 생성하기

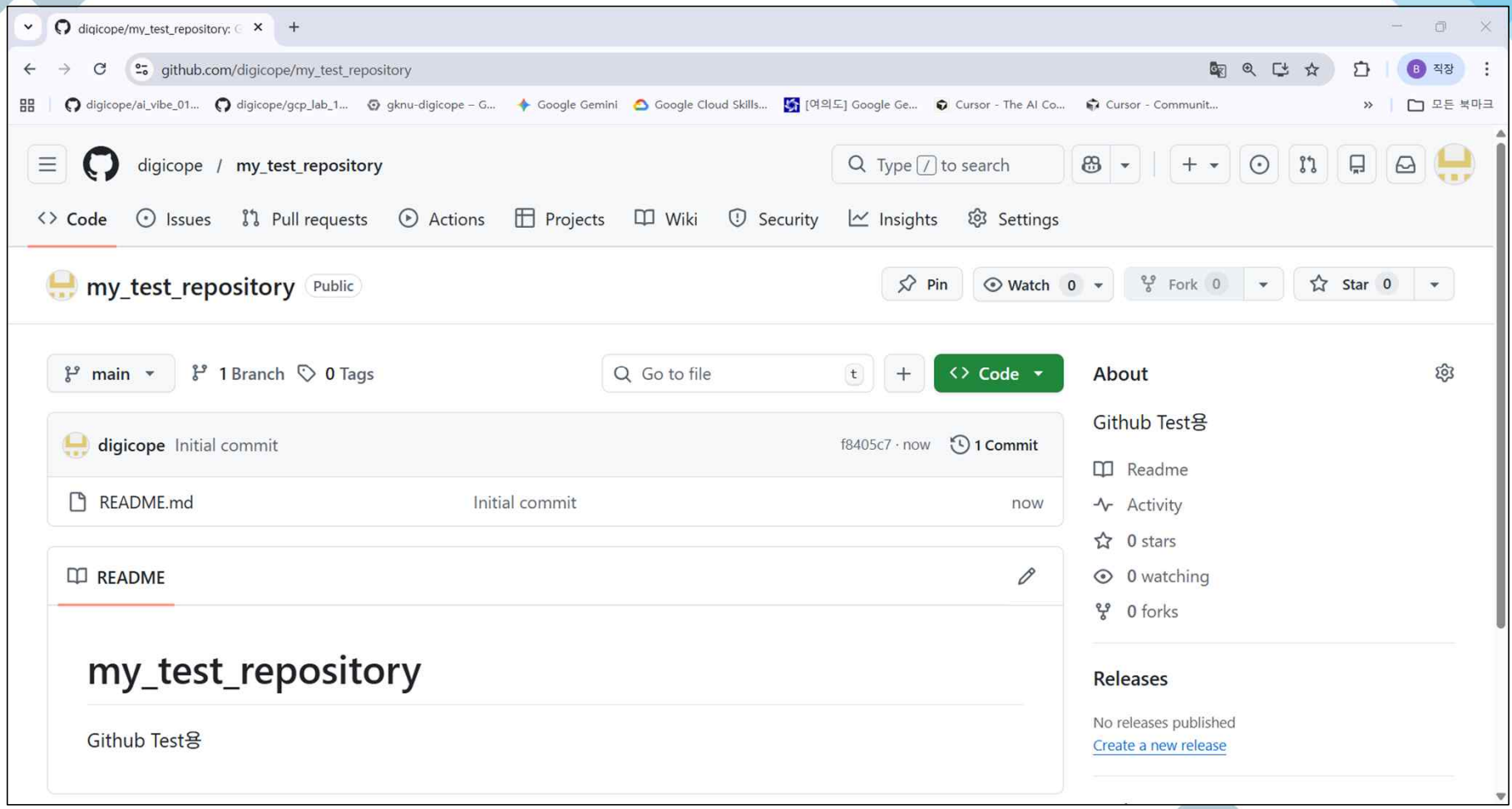
4.1 새 저장소 생성

1. GitHub 로그인 후 오른쪽 상단 + 버튼 클릭
2. New repository 선택

4.2 저장소 설정

- Repository name : 저장소 이름 입력
- Description : 프로젝트 설명 (선택)
- Public : 공개 저장소
- Private : 비공개 저장소
- Add a README file : 체크 권장

Create repository 버튼을 클릭하여 저장소를 생성한다.


digicope/my_test_repository:
github.com/digicope/my_test_repository
digicope / my_test_repository
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
my_test_repository
Public
Pin
Watch 0
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Code
digicope Initial commit
f8405c7 · now
1 Commit
README.md
Initial commit
now
README
my_test_repository
Github Test용
About
Github Test용
Readme
Activity
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release

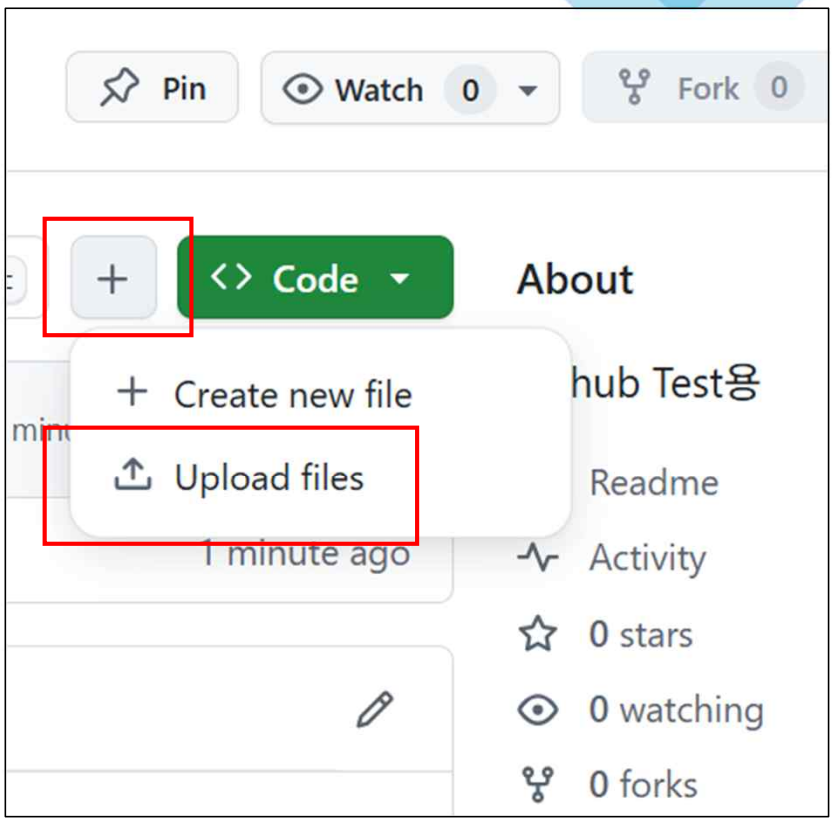
5. 웹에서 파일 업로드하기 (초보자 권장)

5.1 파일 업로드

1. 생성한 저장소로 이동한다.
2. Add file → Upload files 클릭
3. 파일을 드래그하거나 선택하여 업로드한다.
4. Commit message 입력 후 Commit changes 클릭

5.2 README.md 수정

- README.md 파일은 프로젝트 설명 문서이다.
- 연필 아이콘(Edit)을 눌러 내용을 수정할 수 있다.



6. Git 설치 및 기본 사용법 (선택 사항)

6.1 Git 설치

- Windows: https://git-scm.com ↗ 에서 Git for Windows 설치
- 설치 후 터미널(cmd 또는 Git Bash)에서 다음 명령으로 확인한다.

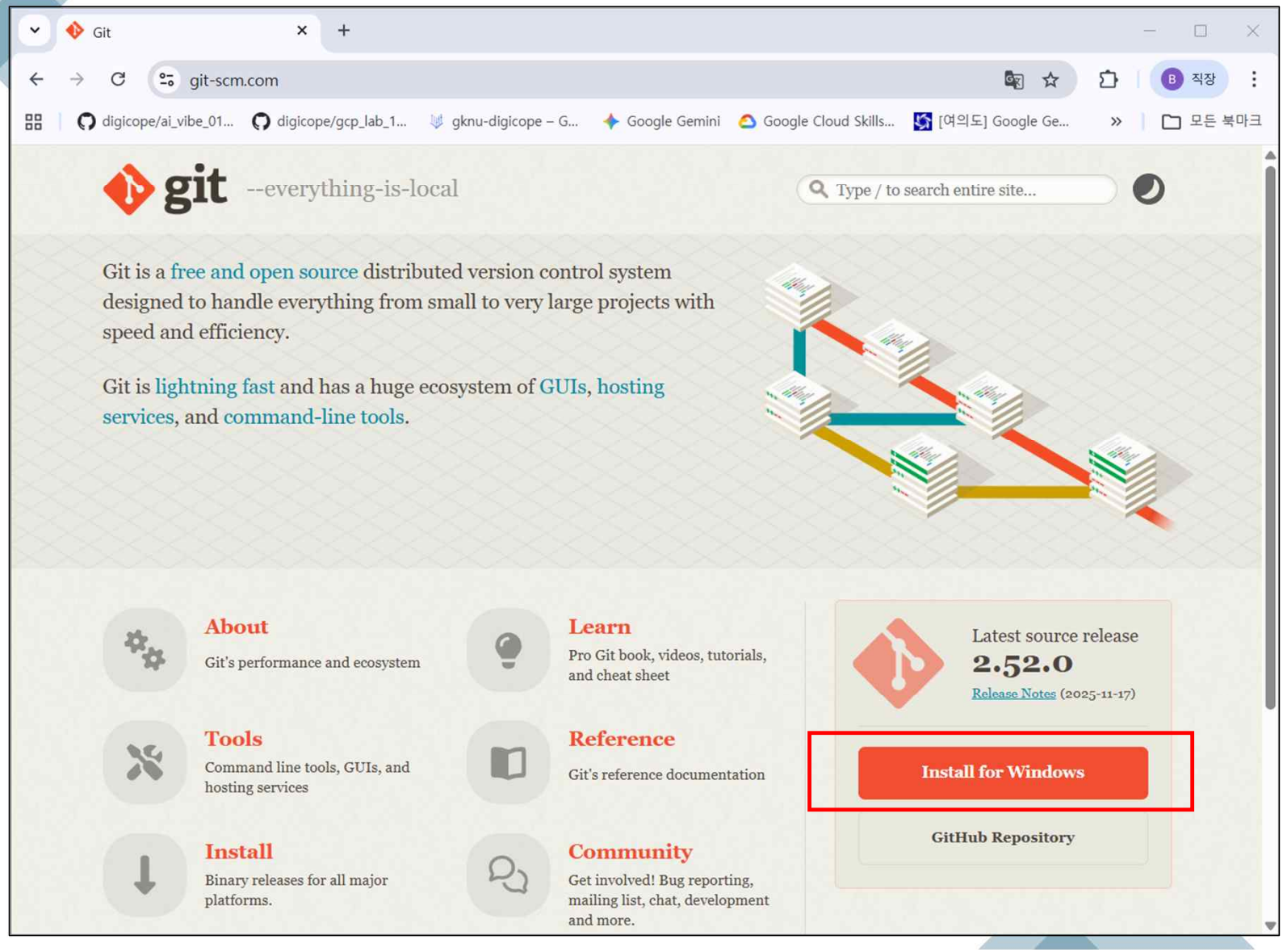
```bash
git --version
```

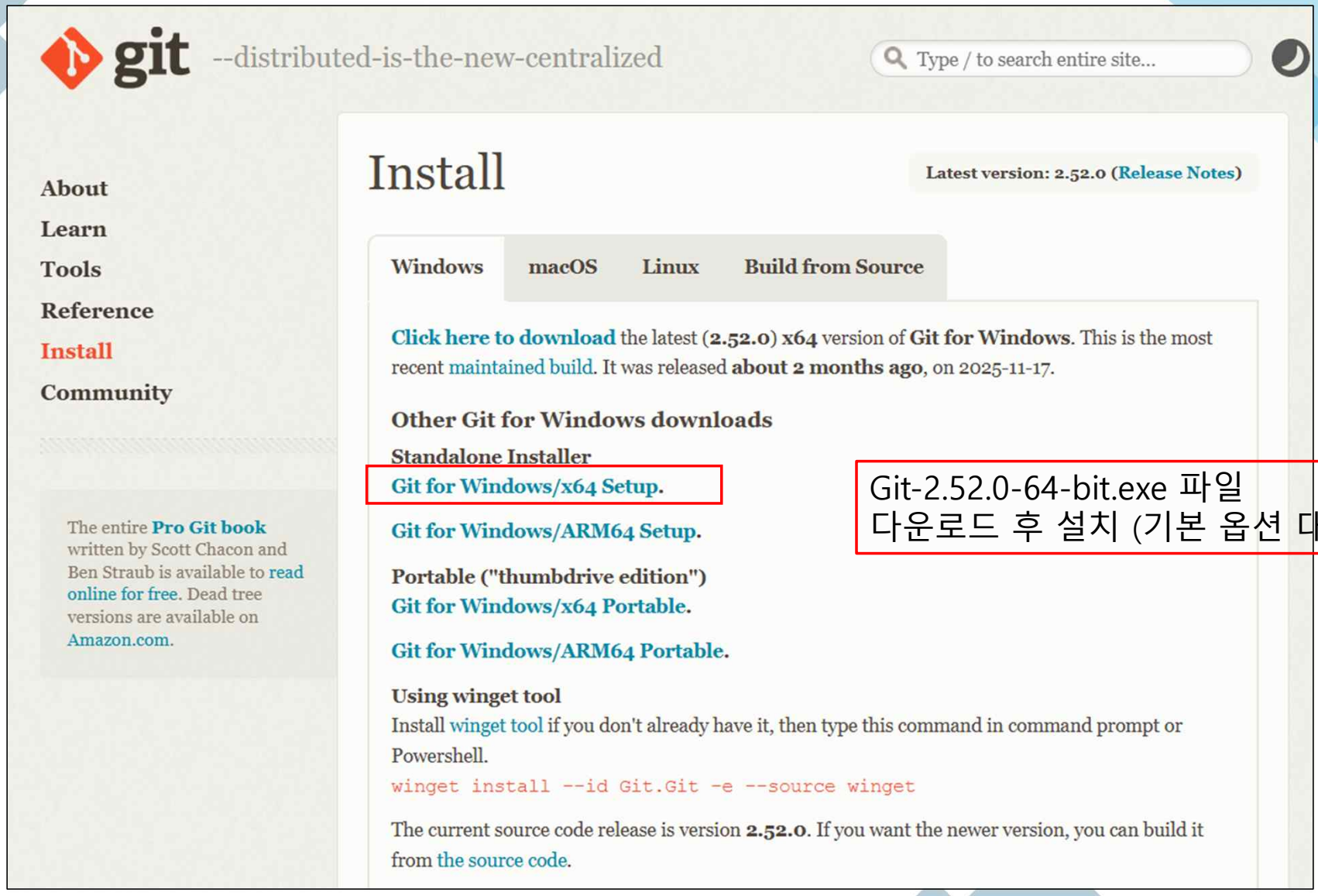
6.2 저장소 복제(Clone)

```bash
git clone https://github.com/사용자이름/저장소이름
```

```
(base) C:\Users\storm\git_test>git clone https://github.com/digicope/ai_vibe_0112
Cloning into 'ai_vibe_0112'...
remote: Enumerating objects: 191, done.
remote: Counting objects: 100% (191/191), done.
remote: Compressing objects: 100% (177/177), done.
Receiving objects:  42% (81/191), 54.58 MiB | 3.54 MiB/s
```

https://git-scm.com/

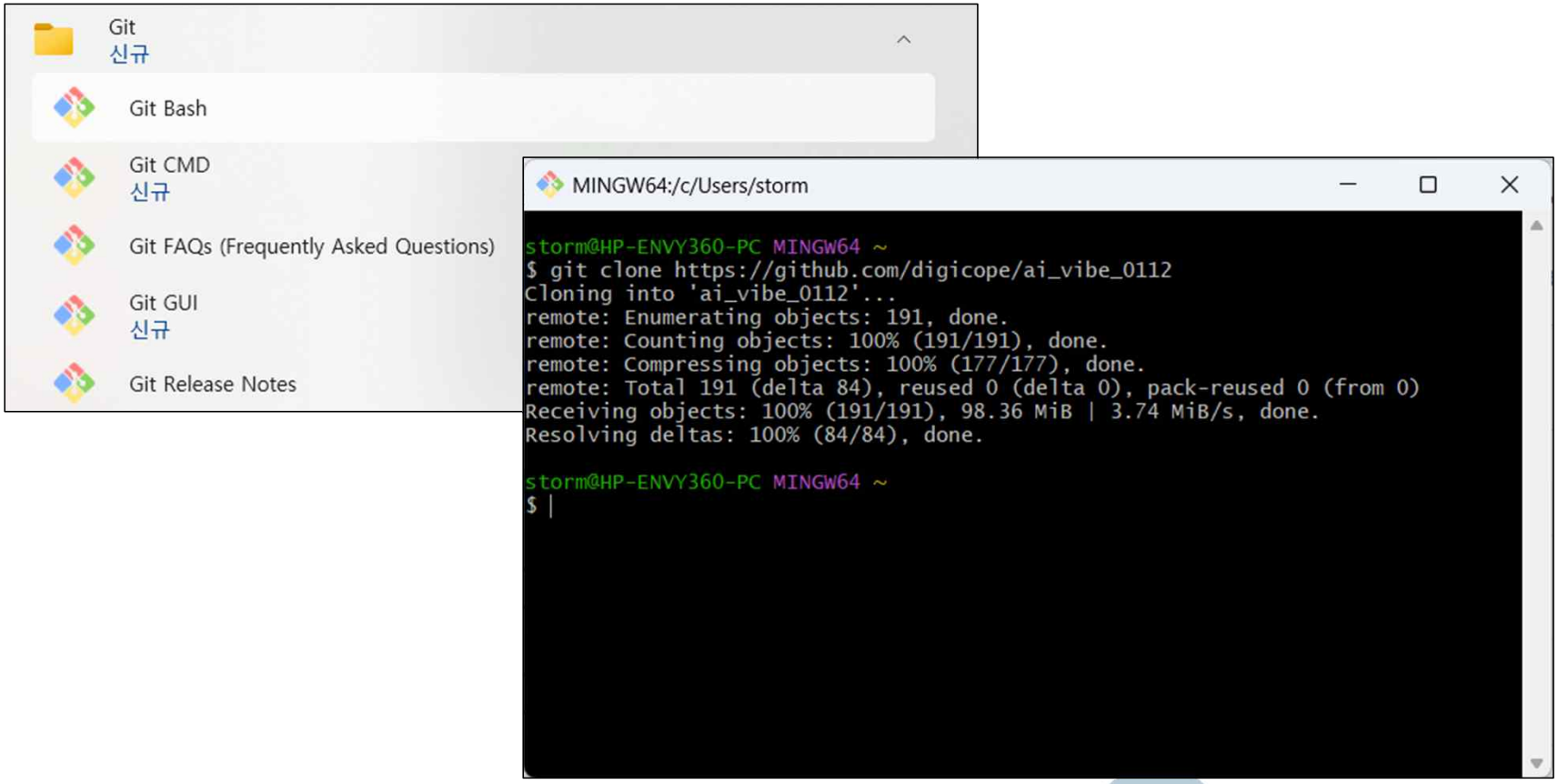




Git-2.52.0-64-bit.exe 파일
다운로드 후 설치 (기본 옵션 대로 설치)

Windows 실행 메뉴에서 [Git Bash]를 실행해서 저장소 복제하기 (전체 가져옴)

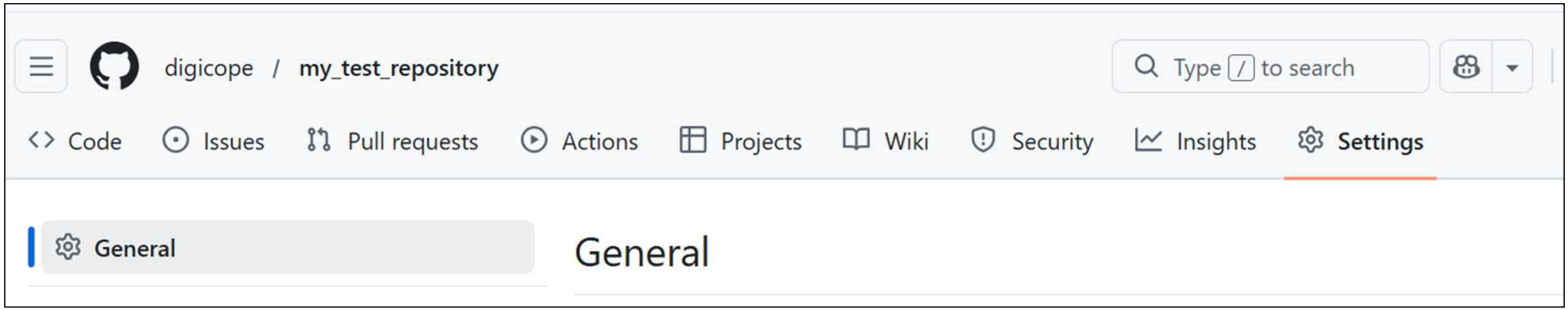
터미널에서 명령 실행 : git clone https://github.com/사용자이름/저장소 이름



레포지토리 삭제하기 (전체 가져옴)

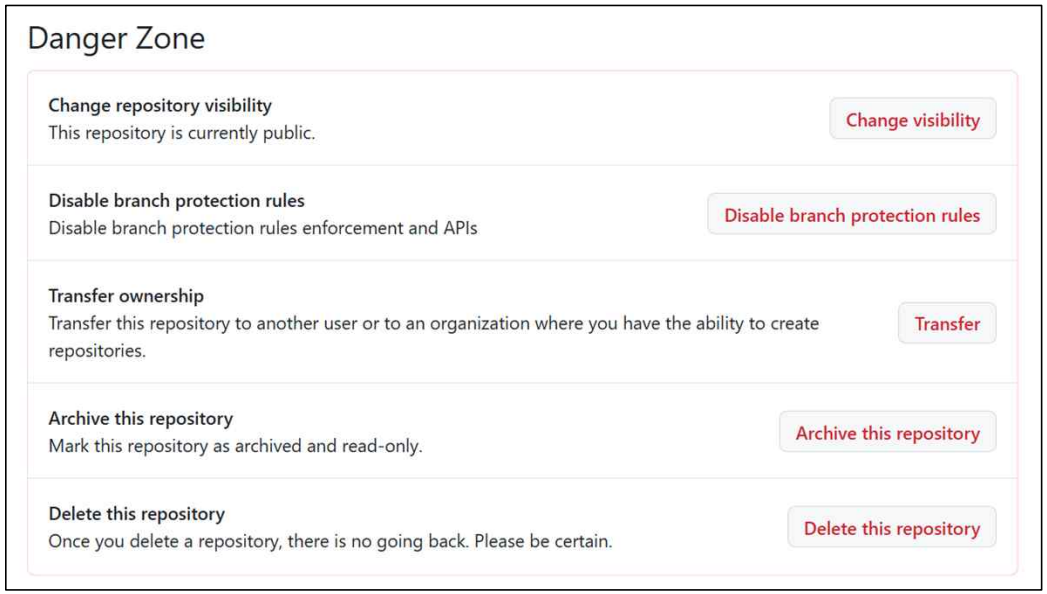
원하는 레포지토리를 클릭하고 우측 상단의 [Settings] 메뉴를 클릭한다

화면 맨아래로 스크롤해서 Danger Zone 으로 내려온다.

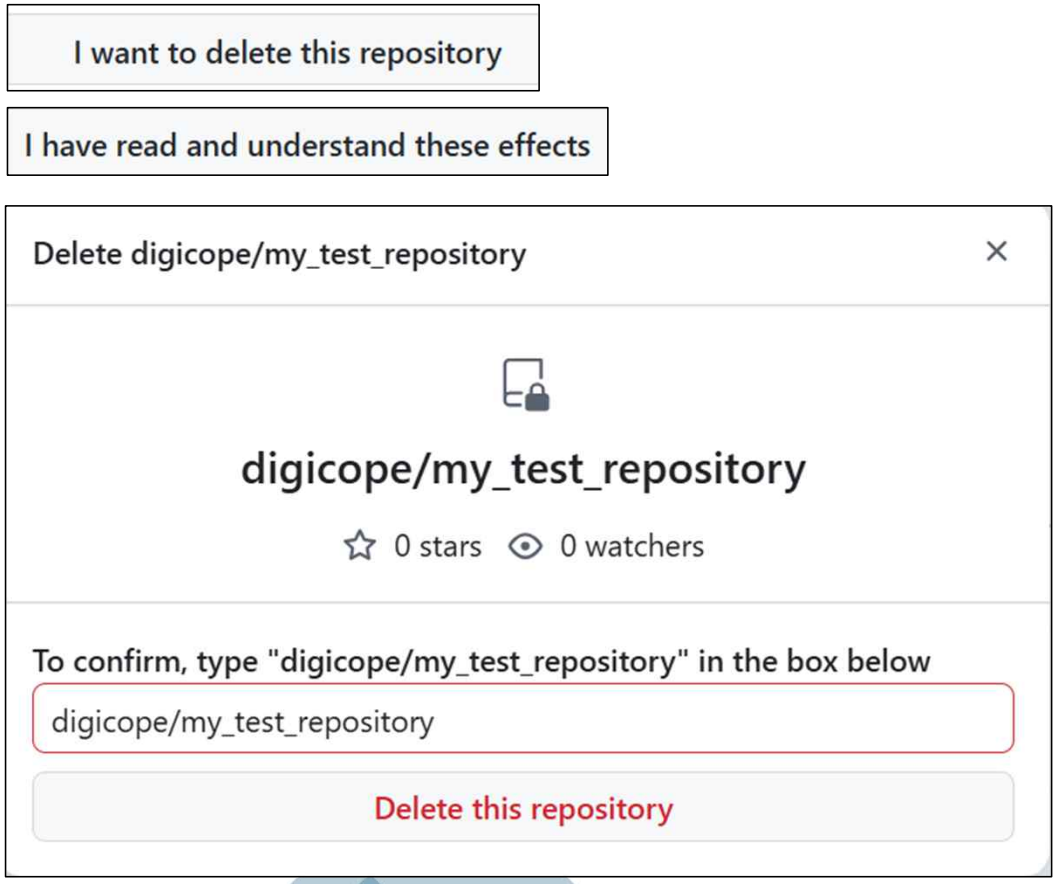


저장소 삭제 방법

Danger Zone에서 **[Delete this repository]** 버튼을 클릭하고 연속 두 번 확인 클릭을 한 다음 아래와 같이 "사용자이름/저장소 이름"을 입력해주고 하단의 **[Delete this repository]** 를 누르면 삭제된다



[Change visibility] 을 누르면 공개로 사용중인 저장소를 private으로 변경하여 비공개로 전환할 수 있다.



3. Github 사용 개발 소스 관리

Git 핵심 용어

용어	쉬운 설명	한 줄 요약
Repository (저장소)	**프로젝트 폴더**	내 코드와 변경 기록이 담기는 "온라인 보관함"입니다.
Commit (커밋)	**세이브 포인트**	"여기까지 저장!" 하고 작업 내용에 이름을 붙여 기록하는 것입니다.
Staging Area (스테이징)	**장바구니**	커밋(저장)하기 전에, 저장할 파일들만 따로 골라 담아두는 곳입니다.
Push (푸시)	**업로드**	내 컴퓨터에서 작업한 '커밋'들을 온라인 GitHub로 보내는 것입니다.
Pull (풀)	**다운로드 & 합치기**	온라인 GitHub에 있는 최신 내용을 내 컴퓨터로 가져와 합치는 것입니다.
Clone (클론)	**통째로 복사**	GitHub에 있는 다른 사람의 프로젝트를 내 컴퓨터로 그대로 복제해 오는 것입니다.
Branch (브랜치)	**나만의 복사본**	원본은 두고, 새로운 기능을 실험해보기 위해 줄기를 나누어 작업하는 것입니다.
Pull Request (PR)	**승인 요청**	"내가 수정한 내용을 원본에 합쳐줘!"라고 관리자에게 제안하는 것입니다.

Windows 실행 메뉴에서 [Git Bash]를 실행하고 터미널에서 아래 명령을 실행한다. (커서의 터미널이나 다른 명령 터미널에서도 가능하다)

(1) 작업에 사용할 폴더를 생성하고 경로를 이동한다

mkdir test_git

cd test_git

(2) Git 초기화 수행 : 이 명령을 실행하면 현재 폴더가 Git 관리 대상이 된다.

git init

(3) 기본 브랜치 이름 변경

git branch -M main

현재 브랜치 이름을 main으로 변경한다

기존에 다른 이름의 브랜치가 있어도 강제로 덮어쓴다

브랜치는 하나의 Git 저장소 안에서 작업 흐름을 분기해서 관리하는 작업 공간이다.

(4) 원격 저장소 등록 (자신의 저장소 주소를 입력하여 사용한다)

git remote add origin https://github.com/사용자이름/저장소이름

git remote add origin https://github.com/digicope/my_test_repository

origin은 원격 저장소의 별칭이다.

하나의 프로젝트에 여러 원격 저장소를 등록할 수도 있다.

원격 저장소 등록 확인

git remote -v

<출력 >

```
$ git remote -v
origin  https://github.com/digicope/my_test_repository (fetch)
origin  https://github.com/digicope/my_test_repository (push)
```

(5) 전역(Global) 사용자 정보 설정

git config --global user.name "본인이름"

git config --global user.email "본인이메일@example.com"

git config --global user.name "digicope"

git config --global user.email digicope@aicore.co.kr

(6) 예제 소스 준비(편집기는 nano 에디터를 사용해도 된다)

touch app.py

echo 'print("Hello GitHub")' > app.py

(7) 소스 올리기 : 파일 스테이징(Stage)

git add app.py → 현재 폴더의 모든 파일을 올리려면 **git add .**

git commit -m "초기 소스 업로드" → -m = message

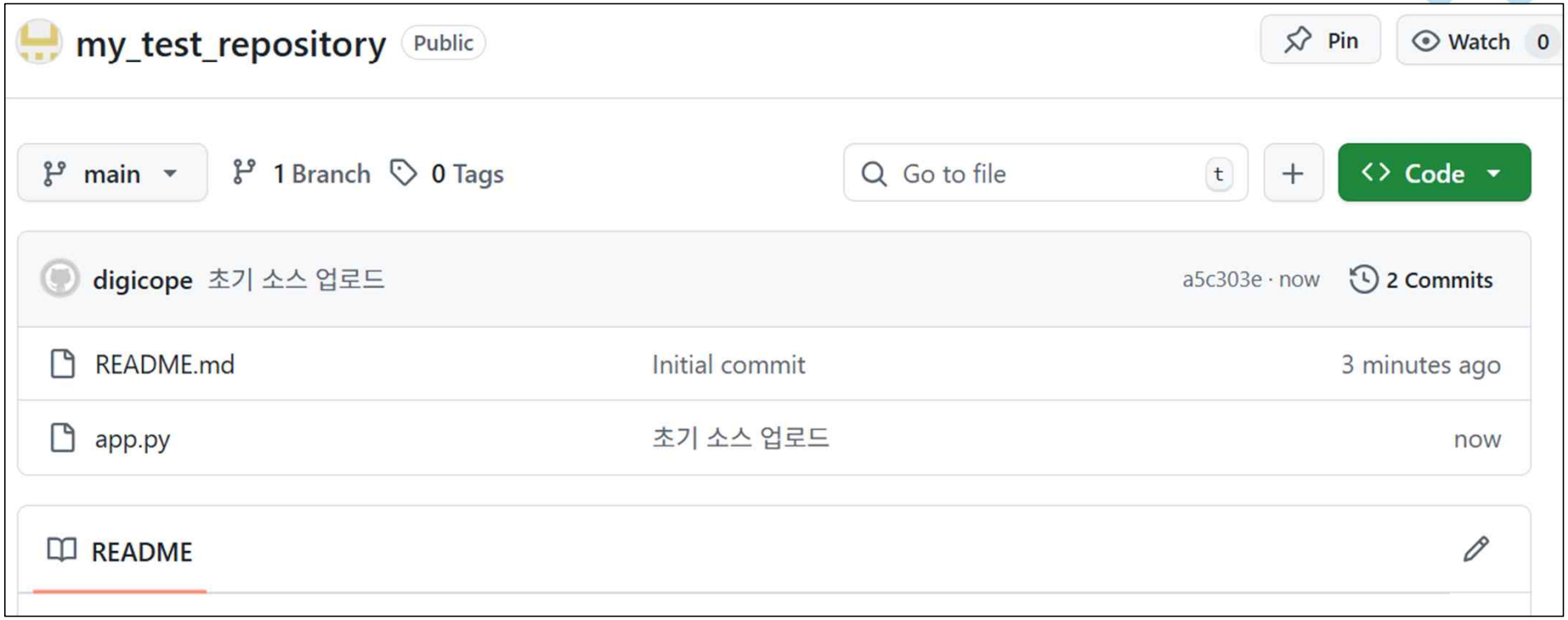
git pull origin main --rebase → 원격에 내 로컬보다 최신 커밋이 있을 때만 필요하다

git push -u origin main → 이때 실제 소스 파일이 업로드 된다.

커밋 명령 설명

명령어	역할	설명
`git add app.py`	스테이징	`app.py` 파일을 커밋 대상(Stage 영역)에 등록한다
`git commit -m "초기 소스 업로드"`	커밋	스테이징된 변경 내용을 하나의 버전으로 Git 저장소에 기록한다
`git pull origin main --rebase`	원격 동기화	원격 저장소(`origin/main`)의 변경 내용을 가져온 뒤, 내 커밋을 그 위에 다시 얹는다
`git push -u origin main`	원격 업로드	로컬 `main` 브랜치의 커밋을 원격 저장소에 업로드한다 (`-u` 는 이후 기본 연결 설정)

소스 파일 업로드 결과 확인



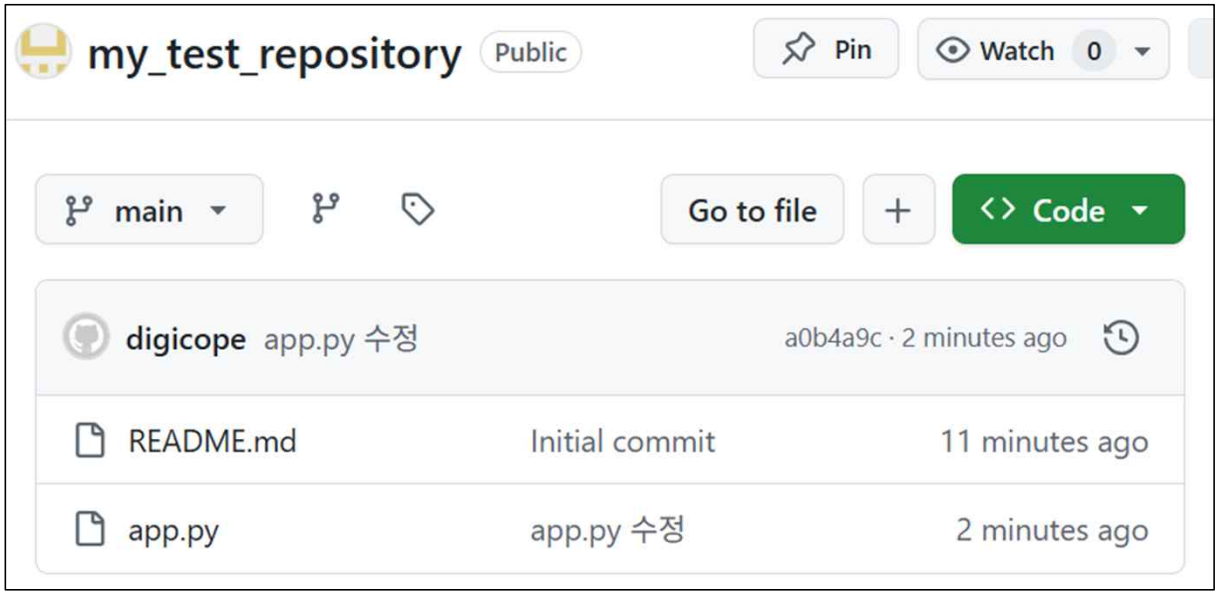
(8) 소스 수정하여 다시 올리기

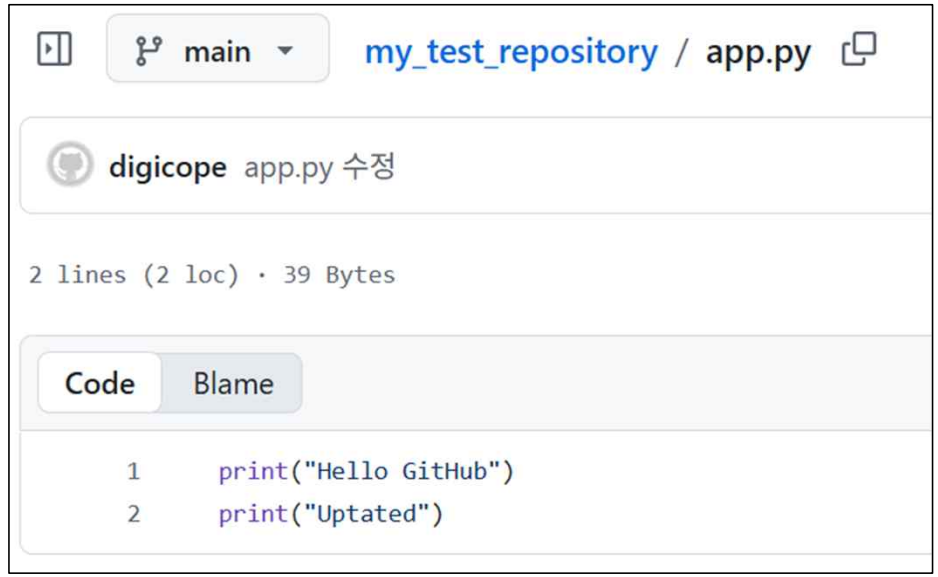
echo 'print("Uptated")' >> app.py

git add app.py

git commit -m "app.py 수정"

git push





(9) 소스 코드 되돌리기

git log --oneline → --oneline의 의미는 커밋 로그를 한 줄씩 간단하게 출력하라는 옵션.

<출력>

a0b4a9c (HEAD -> main, origin/main) app.py 수정

a5c303e 초기 소스 업로드

0b5cab7 Initial commit

- 여기에서 "a5c303e 초기 소스 업로드"로 변경하고 싶은 경우

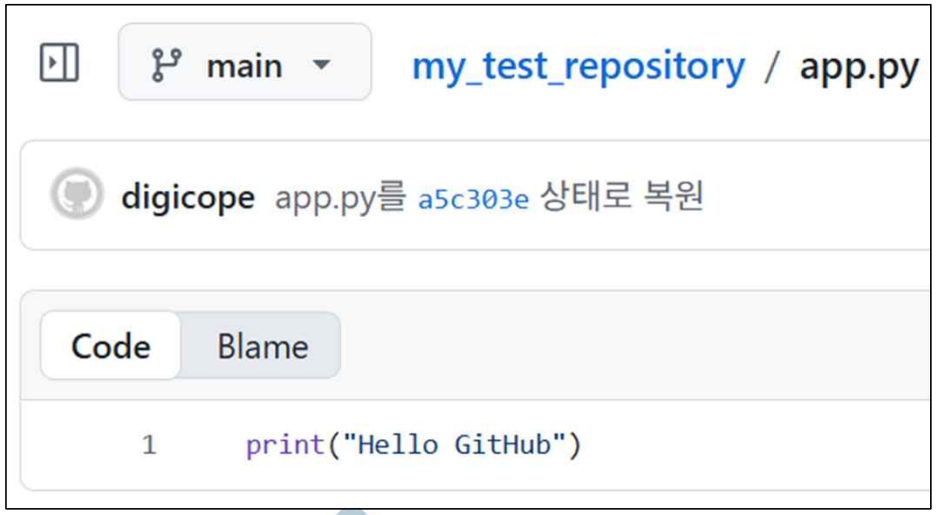
git restore --source=a5c303e app.py

git add app.py

git commit -m "app.py를 a5c303e 상태로 복원"

git push

[결과 확인]



- 다시 "a0b4a9c app.py 수정"으로 변경하고 싶은 경우

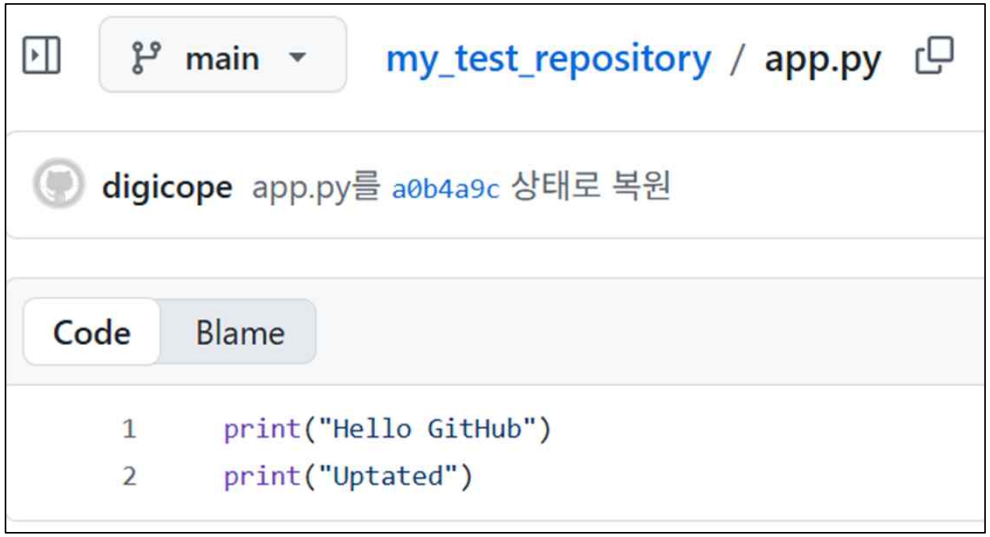
git restore --source=a0b4a9c app.py

git add app.py

git commit -m "app.py를 a0b4a9c 상태로 복원"

git push

[결과 확인]



감사합니다