
2025-가을(게임공) 인공지능

담당교수 : 이재영

완전 초보를 위한 깃허브

https://spartacodingclub.kr/blog/github_guide

<https://sseozytank.tistory.com/41>

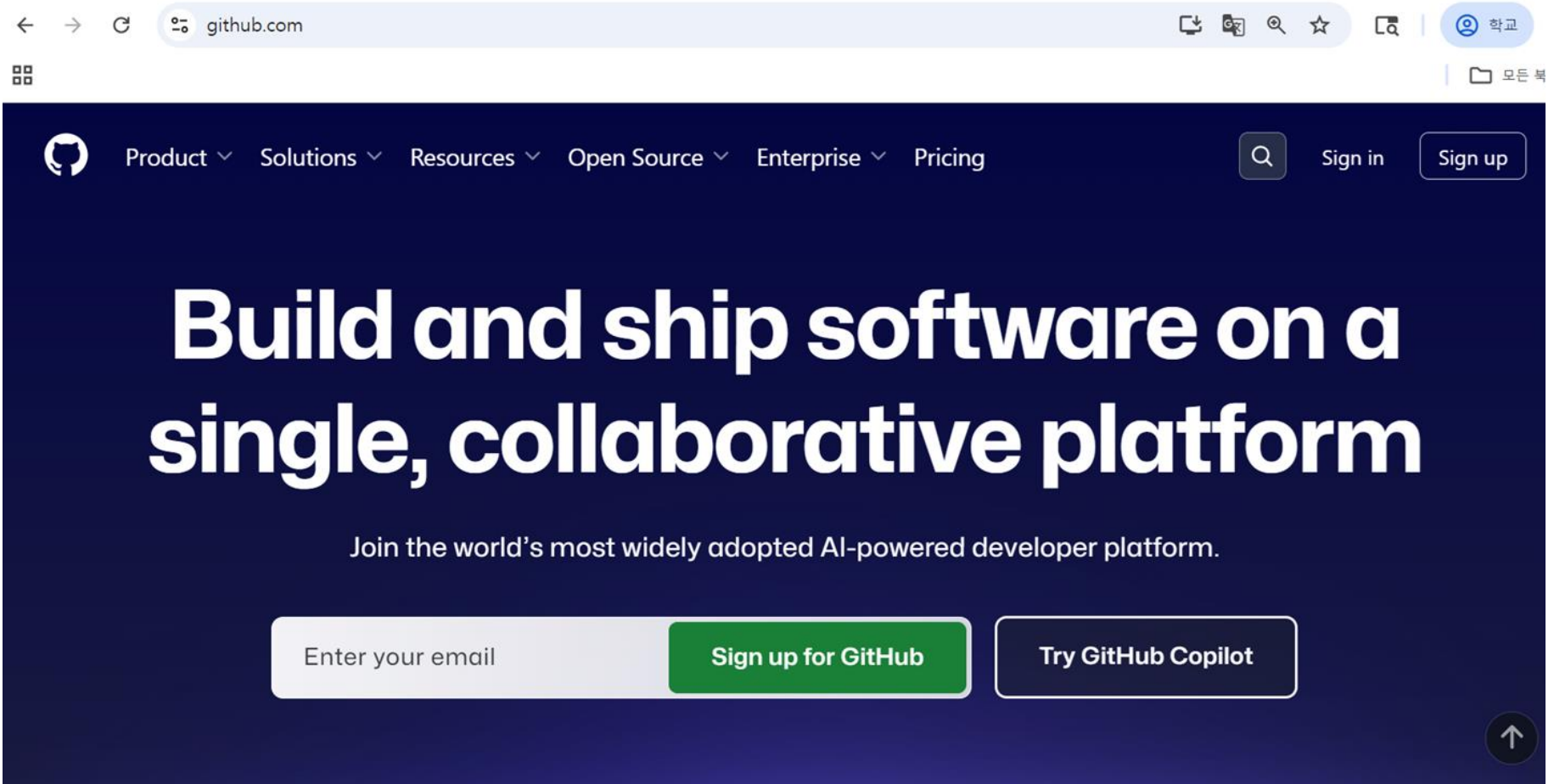
깃이 뭐지?

깃허브의 심장에서 작동되는 소프트웨어인 깃 (Git: 재수없고 멍청한 놈, 자식)을 만든 유명한 소프트웨어 개발자 리누스 토발즈에 감사한다. 깃은 프로젝트의 어떤 부분도 겹쳐쓰지 않게 프로젝트의 변경을 관리하는 버전관리 소프트웨어이다.

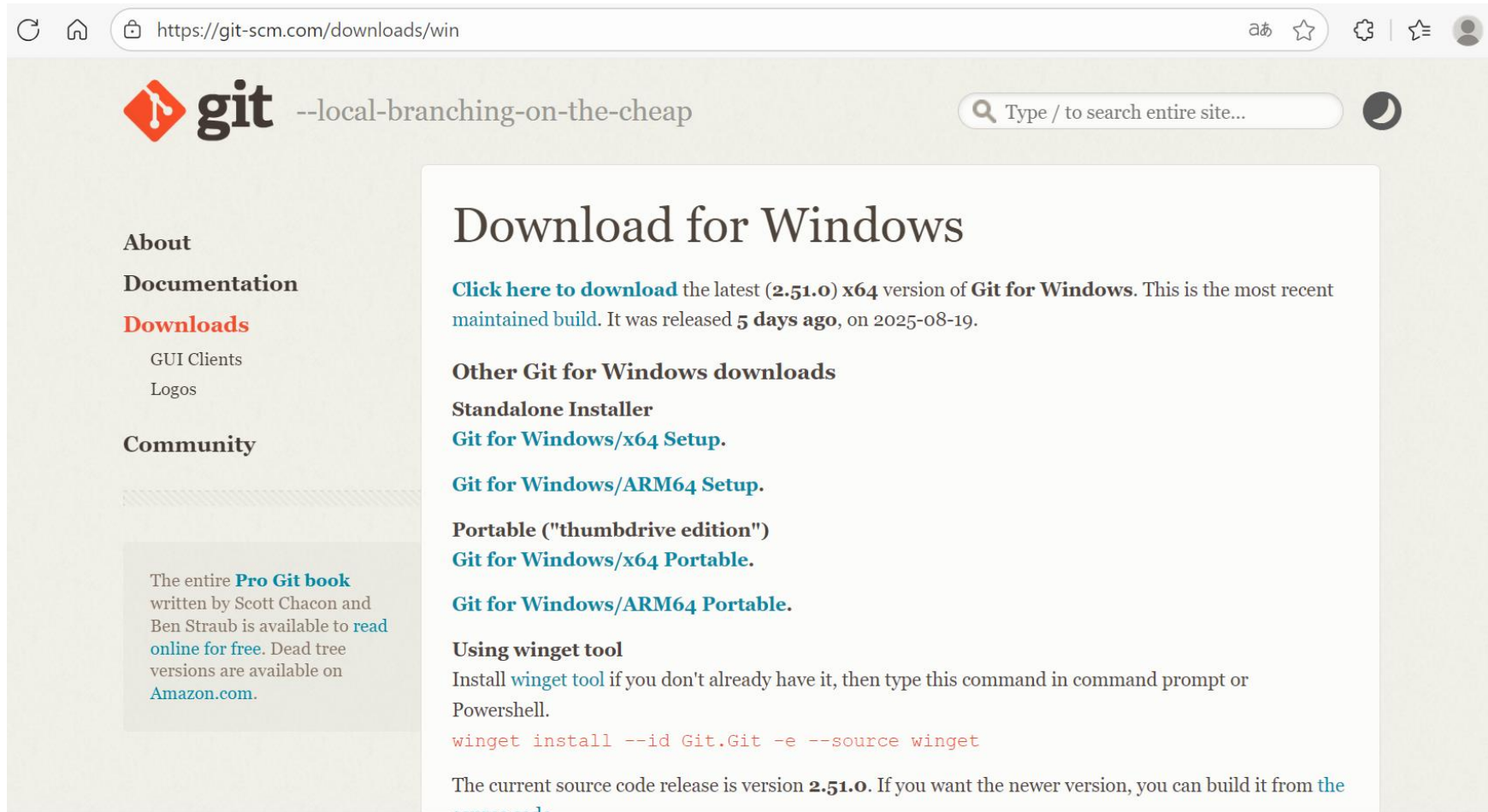
- **커맨드 라인(Command Line):** 깃 명령어를 입력할 때 사용하는 컴퓨터 프로그램. 맥에선 터미널이라고 한다. PC에선 기본적인 프로그램이 아니어서 처음엔 깃을 다운로드해야 한다(다음 섹션에서 다룰 것이다). 두 경우 모두 마우스를 사용하는 것이 아닌 프롬프트로 알려진 텍스트 기반 명령어를 입력한다.
- **저장소(Repository):** 프로젝트가 거주(live)할 수 있는 디렉토리나 저장 공간. 깃허브 사용자는 종종 “repo”로 줄여서 사용한다. 당신의 컴퓨터 안의 로컬 폴더가 될 수도 있고, 깃허브나 다른 온라인 호스트의 저장 공간이 될 수도 있다. 저장소 안에 코드 파일, 텍스트 파일, 이미지 파일을 저장하고, 이름붙일 수 있다.
- **버전관리(Version Control):** 기본적으로, 깃이 서비스되도록 고안된 목적. MS 워드 작업할 때, 저장하면 이전 파일 위에 겹쳐쓰거나 여러 버전으로 나누어 저장한다. 깃을 사용하면 그럴 필요가 없다. 프로젝트 히스토리의 모든 시점의 “스냅샷”을 유지하므로, 결코 잃어버리거나 겹쳐쓰지 않을 수 있다.
- **커밋(Commit):** 깃에게 파워를 주는 명령이다. 커밋하면, 그 시점의 당신의 저장소의 “스냅샷”을 찍어, 프로젝트를 이전의 어떠한 상태로든 재평가하거나 복원할 수 있는 체크포인트를 가질 수 있다.
- **브랜치(Branch):** 여러 명이 하나의 프로젝트에서 깃 없이 작업하는 것이 얼마나 혼란스러울 것인가? 일반적으로, 작업자들은 메인 프로젝트의 브랜치를 따와서 (branch off), 자신이 변경하고 싶은 자신만의 버전을 만든다. 작업4을 끝낸 후, 프로젝트의 메인 디렉토리인 “master”에 브랜치를 다시 “Merge”한다.

GitHub 업로드

- 1단계 github.com 가입하고 온라인 저장소 repository 생성



- 2단계 [Git-2.51.0-64-bit.exe](https://git-scm.com/download/win) 다운로드 및 인스톨 (<https://git-scm.com/download/win>)



https://git-scm.com/downloads/win

git --local-branching-on-the-cheap

Type / to search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (**2.51.0**) **x64** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **5 days ago**, on 2025-08-19.

Other Git for Windows downloads

Standalone Installer

[Git for Windows/x64 Setup.](#)

[Git for Windows/ARM64 Setup.](#)

Portable ("thumbdrive edition")

[Git for Windows/x64 Portable.](#)

[Git for Windows/ARM64 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 **2.51.0**. If you want the newer version, you can build it from [the source code](#).

GitHub 업로드



처음 GitHub 사용을 위한 초기 설정

▶ Git 다운로드 및 설치

```
# Git 버전 확인  
c:\> git --version  
git version 2.51.0.windows.
```

▶ 사용자 정보 설정

```
# 전역 사용자 이름 설정  
c:\> git config --global user.name "홍길동"  
  
# 전역 이메일 설정  
c:\> git config --global user.email hong@example.com  
  
# 설정 확인  
c:\> git config --list user.name=홍길동 user.email=hong@example.com core.autocrlf=true  
  
// Windows에서는 줄바꿈 문자 처리를 위해 core.autocrlf=true 설정이 자동으로 적용함
```

GitHub 업로드

SSH 키 설정 및 GitHub 연결

SSH 키 생성

```
# SSH 키 생성 (RSA 4096비트)
c:\> ssh-keygen -t rsa -b 4096 -C "hong@example.com"
// Generating public/private rsa key pair. Enter file in which to save the key
(C:\Users\User\.ssh\id_rsa): [Enter]
Enter passphrase (empty for no passphrase): [Enter]
Enter same passphrase again: [Enter]
Your identification has been saved in C:\Users\User\.ssh\id_rsa
Your public key has been saved in C:\Users\User\.ssh\id_rsa.pub
```

SSH 에이전트 시작 및 키 등록

```
# SSH 에이전트 시작
c:\> eval $(ssh-agent -s)
Agent pid 2048

# SSH 키 등록
c:\> ssh-add ~/.ssh/id_rsa
Identity added: C:\Users\User\.ssh\id_rsa (hong@example.com)
```

GitHub 업로드

공개 키 확인 및 GitHub 등록

공개 키 내용 확인

```
c:\> cat ~/.ssh/id_rsa.pub ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQC...  
                                hong@example.com
```

GitHub 연결 테스트

```
c:\> ssh -T git@github.com
```

```
Hi 사용자명! You've successfully authenticated, but GitHub does not provide shell access.
```


GitHub 업로드



공동 작업 환경 구성

▶ 리포지토리(Repository) 생성 및 초기화

로컬 프로젝트 폴더 생성

```
# 프로젝트 폴더 생성  
c:\> mkdir team-project  
c:\> cd team-project
```

```
# Git 저장소 초기화  
c:\team-project>git init // Initialized empty Git repository in C:/team-project/.git/
```

README 파일 생성 및 첫 커밋(commit)

README 파일 생성

```
c:\team-project> echo "# Team Project" > README.md
```

```
c:\team-project> echo "팀 프로젝트 협업 저장소입니다." >> README.md
```

파일 스테이징

```
c:\team-project> git add README.md
```

첫 번째 커밋

```
c:\team-project> git commit -m "Initial commit: Add README"
```

```
[main (root-commit) a1b2c3d] Initial commit: Add README 1 file changed,  
2 insertions(+) create mode 100644 README.md
```

GitHub 원격 저장소 연결

원격 저장소 추가

```
c:\team-project> git remote add origin
```

```
c:\team-project> git@github.com:username/team-project.git
```

원격 저장소에 푸시(push)

```
c:\team-project> git branch -M main git push -u origin main
```

```
Enumerating objects: 3, done. Counting objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 267 bytes | 267.00 KiB/s, done. Total 3 (delta 0),
```

```
reused 0 (delta 0), pack-reused 0 To github.com:username/team-project.git *
```

```
[new branch] main -> main Branch 'main' set up to track remote branch 'main' from 'origin'.
```

GitHub 업로드

브랜치 전략 및 팀 협업 워크플로우



▶ 브랜치 생성 및 관리

개발 브랜치 생성

```
c:\> git checkout -b develop // Switched to a new branch 'develop'
```

기능 브랜치 생성

```
c:\> git checkout -b feature/user-login // Switched to a new branch 'feature/user-login'
```

브랜치 목록 확인

```
c:\> git branch -a develop * feature/user-login main remotes/origin/main
```

▶ 팀원과의 협업 과정

작업 후 커밋

```
c:\> git add login.js git commit -m "feat: Add user login functionality"
```

원격 저장소에 브랜치 푸시

```
c:\> git push -u origin feature/user-login
```

```
// Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 To
```

```
// github.com:username/team-project.git * [new branch] feature/user-login ->
```

```
// feature/user-login Branch 'feature/user-login' set up to track
```

```
// remote branch 'feature/user-login' from 'origin'.
```

병합 및 충돌 해결

▶ Pull Request 후 병합

```
# develop 브랜치로 전환
c:\> git checkout develop

# 최신 변경사항 가져오기
c:\> git pull origin develop // From github.com:username/team-project * branch develop ->
                               // FETCH_HEAD Already up to date.

# 기능 브랜치 병합
c:\> git merge feature/user-login // Updating a1b2c3d..e4f5g6h Fast-forward login.js | 25
                                   // 1 file changed, 25 insertions(+)
```

▶ 충돌 해결 과정

충돌 발생 시

```
c:\> git merge feature/user-auth // Auto-merging login.js CONFLICT (content): Merge conflict
// in login.js Automatic merge failed; fix conflicts and then
// commit the result.
```

충돌 파일 확인

```
c:\> git status // On branch develop You have unmerged paths.
// (fix conflicts and run "git commit")
// (use "git merge --abort" to abort the merge)
// Unmerged paths: (use "git add ..." to mark resolution)
// both modified: login.js
```

충돌 해결 후 커밋

```
c:\> git add login.js
c:\> git commit -m "resolve: Fix merge conflict in login.js"
// [develop i7j8k9l] resolve: Fix merge conflict in login.js
```

자주 사용하는 Git 명령어 모음

명령어	설명	사용 예시
<code>git status</code>	작업 디렉토리 상태 확인	변경된 파일, 스테이징된 파일 확인
<code>git log --oneline</code>	커밋 히스토리 간략 조회	최근 커밋들을 한 줄로 표시
<code>git diff</code>	변경사항 비교	스테이징 전 변경내용 확인
<code>git reset --soft HEAD~1</code>	마지막 커밋 취소 (변경사항 유지)	커밋 메시지 수정할 때
<code>git stash</code>	임시 작업 저장	브랜치 전환 전 작업 보관
<code>git cherry-pick <commit></code>	특정 커밋만 가져오기	다른 브랜치의 특정 변경사항 적용

Pro Tips:

- `git config --global alias.st status`로 단축 명령어 설정
- `git log --graph --oneline --all`로 브랜치 시각화
- `.gitignore` 파일로 불필요한 파일 제외
- 커밋 메시지는 "타입: 설명" 형식으로 작성 (feat, fix, docs, style, refactor, test, chore)

GitHub 명령어

- 로컬저장소에서 **Git-CMD 실행 예 (작업 폴더) commit & push**

```
git config --global user.name "이름"
git config --global user.email "깃허브 메일주소" // 매번 물어보는 귀찮음을

mkdir ~/MyProject // 로컬 디렉토리 만들고
cd ~/myproject // 디렉토리로 들어가서
git init // 깃 명령어를 사용할 수 있는 디렉토리로 만든다.
git status // 현재 상태를 훑어보고
git add 파일명.확장자 // 깃 주목 리스트에 파일을 추가하고 or
git add . // 이 명령은 현재 디렉토리의 모든 파일을 추가할 수 있다.
git commit -m "현재형으로 설명" // 커밋해서 스냅샷을 찍는다.

git remote add origin https://github.com/username/myproject.git // 로
git remote -v // 연결상태를 확인한다.
git push origin master // 깃허브로 푸시한다.
```

GitHub 업로드

- 흔히 하는 예러
 - 타이핑 예러, V3 방화벽 차단, 너무 많은 파일과 디렉터리 커밋
 - 해결 : `rmdir .git /S` (.git 디렉터리 삭제한 후 처음 부터 다시 실행)
- **[rejected] master -> master (fetch first) 예러**
 - 해결1 : `git pull` 실행 후 `git push origin master` 실행
 - 해결2 : `git push origin master -f` (강제로 실행)

```
Git CMD
C:\Users\kys>git config --global user.name "Mayfifth"
C:\Users\kys>git config --global user.email "kys@kpu.ac.kr"
C:\Users\kys>git init
Initialized empty Git repository in C:/Users/kys/.git/
C:\Users\kys>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .appcfg_oauth2_tokens
        .bash_history
        .gitconfig
```

GitHub 업로드

Git CMD

```
ntuser.ini
socket_client.py
socket_server.py
temp.py
test.txt
```

nothing added to commit but untracked

```
C:\Users\kys>git add temp.py
```

```
C:\Users\kys>git commit -m "first commit"
[master (root-commit) 5eb09d0] first commit
1 file changed, 25 insertions(+)
create mode 100644 temp.py
```

```
C:\Users\kys>git remote add origin https://github.com/Mayfifth/book-manager.git
```

```
C:\Users\kys>git remote -v
origin https://github.com/Mayfifth/book-manager.git (fetch)
origin https://github.com/Mayfifth/book-manager.git (push)
```

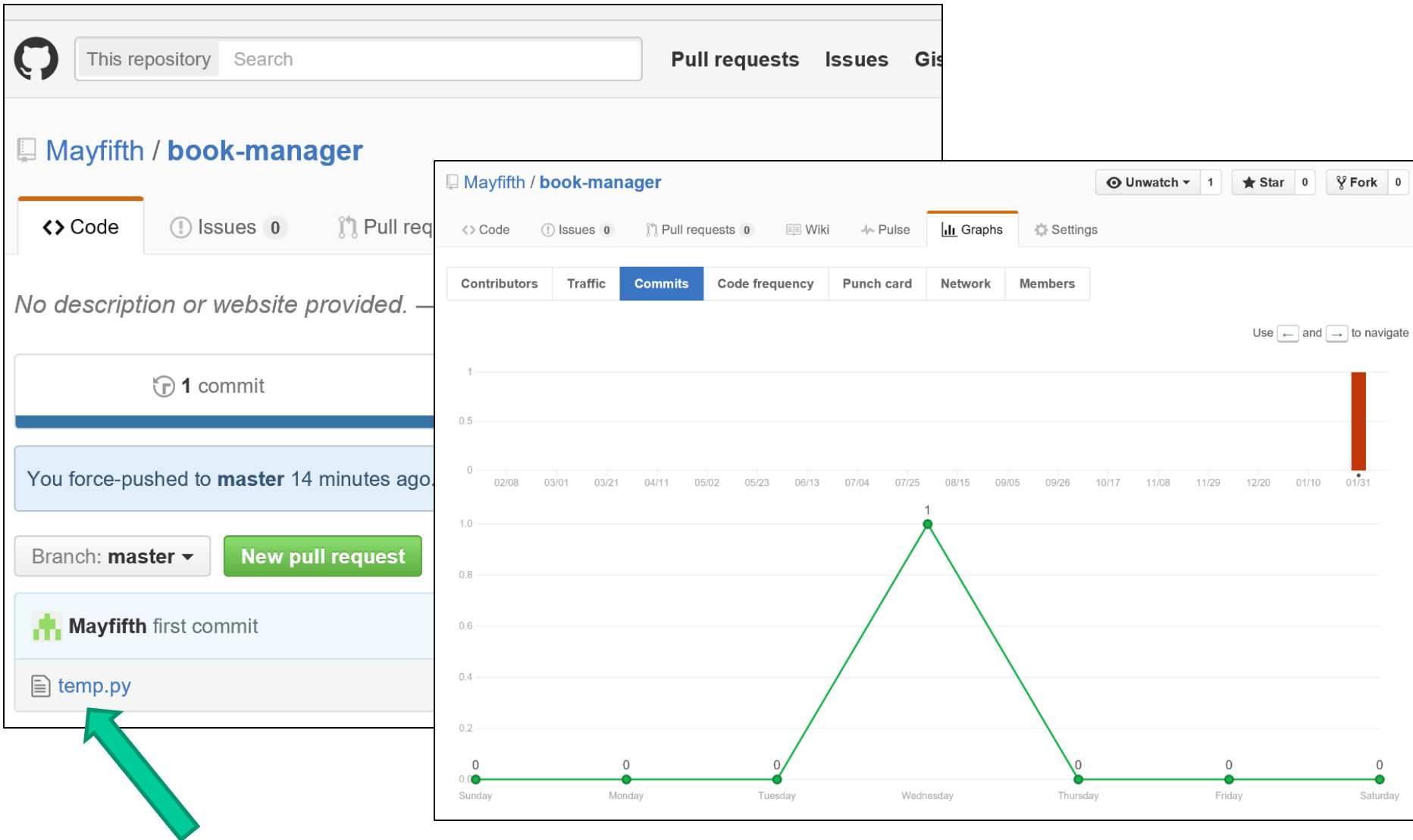
```
C:\Users\kys>git push origin master
Username for 'https://github.com': Mayfifth
Password for 'https://Mayfifth@github.com':
To https://github.com/Mayfifth/book-manager.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Mayfifth/book-manager.git'
```

```
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Git CMD

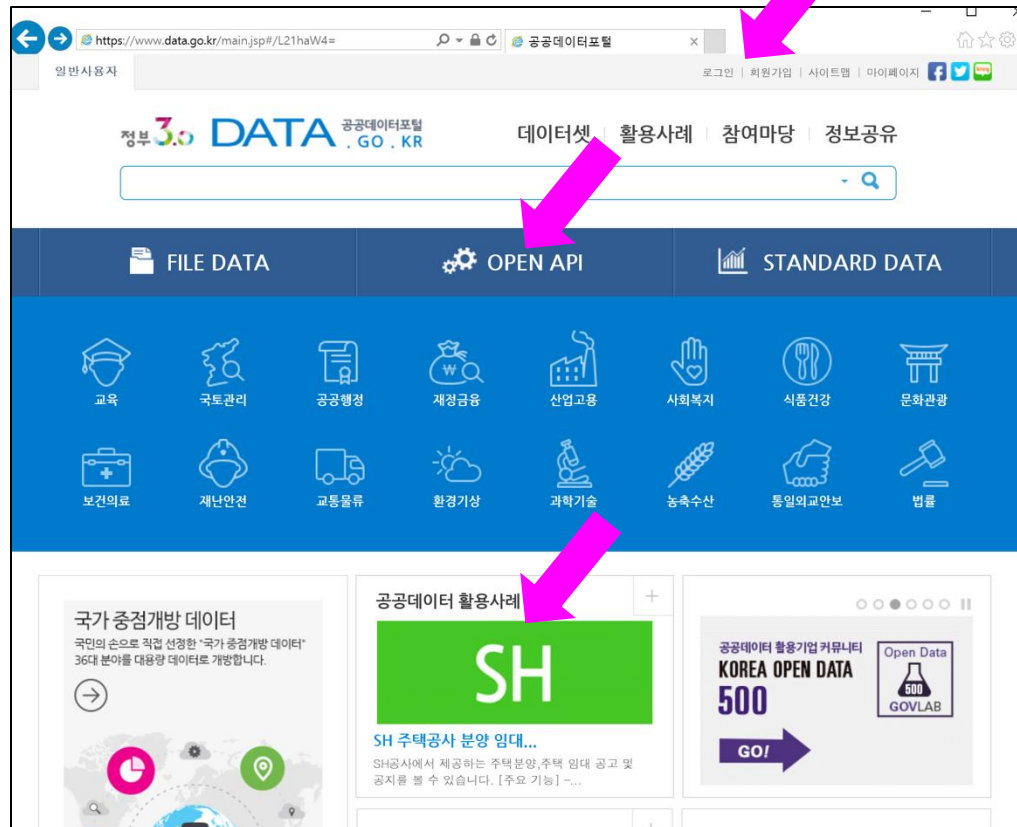
```
C:\Users\kys>git push origin master -f
Username for 'https://github.com': Mayfifth
Password for 'https://Mayfifth@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 436 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Mayfifth/book-manager.git
+ b0d5ba0...5eb69d0 master -> master (forced update)
```

GitHub 업로드



국가 공공 데이터 포털 (<http://www.data.go.kr>)

- 공공데이터 OpenAPI 국회입법정보
 - 국회 의원정보, 회의록, 의사일정, 국회의원 정보 등 입법관련 정보를 민간 및 공공기관에서 활용할 수 있도록 표준화 방식으로 개방한 공유 서비스



국가 공공 데이터 포털 (<http://www.data.go.kr>)

■ OpenAPI 통합검색

The screenshot shows the National Public Data Portal interface. At the top, there's a navigation bar with the 'DATA GO.KR' logo and links for '데이터셋', '활용사례', '참여마당', '정보공유', and '전체메뉴'. Below this is a search bar with a magnifying glass icon. A pink arrow points to the search bar, which contains the text '통합검색' and a dropdown menu showing '국회의원'. Below the search bar, there are tabs for '전체(5)', '파일데이터(0)', '오픈API(3)', '표준데이터(0)', and '게시물(2)'. A button labeled '초기화' is also present. The main content area displays the message '오픈API 3건을 찾았습니다.' (Found 3 OpenAPIs). Below this, there's a blue header for '오픈API [3건]' with filters for '정확도순', '날짜순', '제목순', '조회순', and '활용신청'. Two results are listed: '국회의원 정보' and '의사일정 정보'. Each result includes details like '조회수', '활용신청건수', '수정일', '기관', '서비스유형', and a 'XML' button. A pink arrow points to the '오픈API [3건]' header. On the right side, there's a sidebar with '인기검색어' (Popular Search Terms) and a list of categories like '경제', '지하철', '버스', '날씨', and '우편번호'. Below this is a section for '국가중점데이터' (National Key Data) with expandable categories like '서비스유형필터(OPENAPI)', '제공기관필터', and '분류체계필터'. Two orange arrows labeled '다운로드' (Download) point to the 'XML' buttons of the search results.

국가 공공 데이터 포털 (<http://www.data.go.kr>)

■ 활용 신청

국가 공공 데이터 포털 (DATA GO.KR)

데이터셋 | 활용사례 | 참여마당 | 정보

국회의원 정보
(국회입법정보) 국회의원 상세정보 및 소속정당, 지역정보 조회

활용신청 건수(바로그가 횡수) : 24

국회의원 정보제공 서비스

Q 연관 데이터셋

연관데이터가 없습니다.

개발계정 신청

OPEN API > 개발계정 신청

기본정보

서비스명	장기요양기관 시설·상세정보조회 서비스	서비스 유형	REST
실의여부	자동승인	신청유형	개발계정 활용신청
처리상태	신청	활용기간	승인일로부터 24개월 간 활용가능

시스템유형 선택

일반 ☒ 서버 구축

시스템 유형

- 일반 : OpenAPI 서비스를 호출하여 응답받은 결과값을 서버에 저장하지 않고 사용할 경우 (서버 미구축)
- 서버 구축 : OpenAPI 서비스를 호출하여 응답받은 결과값을 서버에 저장하거나 DB화 하여 사용할 경우

안내

위치가반서비스 사업자 확인

공공데이터를 위치정보를 포함한 서비스를 사용하고자 하는 사업자는 '위치정보의 보호 및 이용 등에 관한 법률'에 따라 방송통신위원회에 '위치정보서비스 허가'를 받거나 '위치가반 서비스사업 신고'를 하여야 합니다. 이에 해당하는 사업자인 경우에는 첨부파일에 '위치가반서비스사업신고필증'을 첨부하여 주시기 바랍니다. 활용신청 시 '위치가반서비스사업신고필증'이 등록되지 않으면 반려가 될 수 있습니다. 신고하시기 바랍니다.

활용정보

활용목적

☒ 웹 사이트 개발 ☐ 앱개발 (모바일, 스마트 등) ☐ 기타 ☐ 참고자료 ☐ 연구(논문 등)

※파일 첨부시 파일첨단 기능이 해제되어야 합니다.

첨부파일 추가 삭제 한 개의 파일만 첨부할 수 있습니다.

상세기능정보 필수 입력 정보입니다.

*자동승인 상세기능은 신청과 동시에 활용 가능합니다.

<input type="checkbox"/>	상세기능	설명	발달 트래픽
<input type="checkbox"/>	기관기타 상세 정보조회	기관기타 상세 정보조회	1000
<input type="checkbox"/>	수용인원 상세 정보조회	수용인원 상세 정보조회	1000
<input type="checkbox"/>	복지종구 현황 상세 정보조회	복지종구 현황 상세 정보조회	1000
<input type="checkbox"/>	협약기관 현황 상세 정보조회	협약기관 현황 상세 정보조회	1000
<input type="checkbox"/>	프로그램현황 상세 정보조회	프로그램현황 상세 정보조회	1000
<input type="checkbox"/>	비급여현황 상세 정보조회	비급여현황 상세 정보조회	1000
<input type="checkbox"/>	시설현황 상세 정보조회	시설현황 상세 정보조회	1000
<input type="checkbox"/>	인력현황 상세 정보조회	인력현황 상세 정보조회	1000
<input type="checkbox"/>	일반현황 상세 정보조회	일반현황 상세 정보조회	1000

라이선스표시

이용여력범위 없음

(사유 :)

신청 취소

국가 공공 데이터 포털 (<http://www.data.go.kr>)

■ API 키 발급

마이페이지

오픈API

- 개발계정
- 활용현황
- 운영계정
- 인증키 발급현황

DATA

나의 문의

나의 관심

회원정보

개발계정

신청 0건

활용 1건

중지 0건

인증키 발급현황

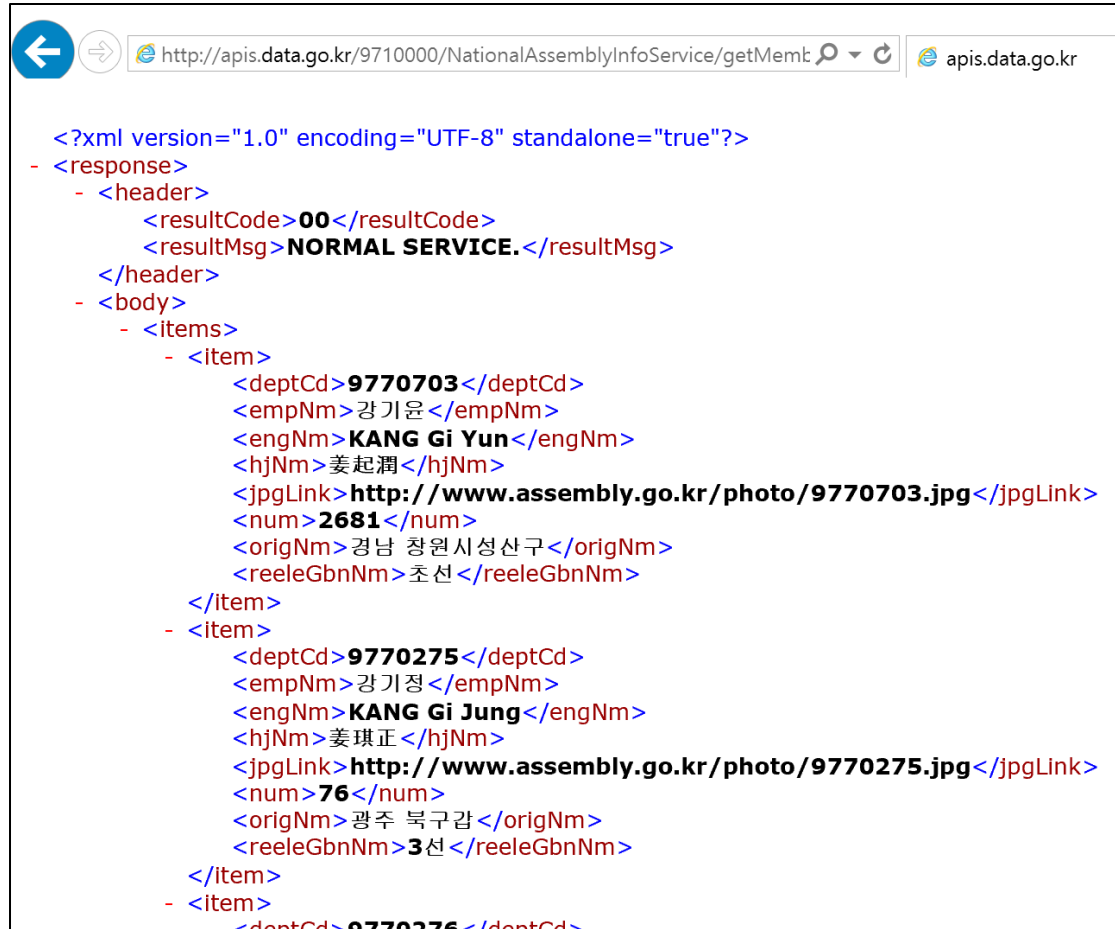
구분	발급일자	발급시간	재발급여부	인증키
일반용	2016/01/31	22:13:07	신규	sea100UMmw23Xyys33F1EQnumONR%2F9ElxBLzkiU9Yr1oT4TrCot8Y2p0jyUJ P72x9rG9D8CN5yuEs6AS2sAiw%3D%3D

인증키 발급현황				
총 1 건				
전체 ▼ 검색 🔍				
구분	발급일자	발급시간	재발급여부	인증키
일반용	2016/01/31	22:13:07	신규	sea100UMmw23Xyys33F1EQnumONR%2F9ElxBLzkiU9Yr1oT4TrCot8Y2p0jyUJ P72x9rG9D8CN5yuEs6AS2sAiw%3D%3D

국가 공공 데이터 포털 (<http://www.data.go.kr>)

■ 공공데이터 OpenAPI 국회입법정보 URL 예시

- <http://apis.data.go.kr/9710000/NationalAssemblyInfoService/getMemberCurrStateList?ServiceKey=sea100UMmw23Xycs33F1EQnumONR%2F9ElxBLzkilU9Yr1oT4TrCot8Y2p0jyuJP72x9rG9D8CN5yuEs6AS2sAiw%3D%3D>



The screenshot shows a web browser window with the URL <http://apis.data.go.kr/9710000/NationalAssemblyInfoService/getMemt>. The browser displays an XML response. The XML structure is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <response>
  - <header>
    <resultCode>00</resultCode>
    <resultMsg>NORMAL SERVICE.</resultMsg>
  </header>
  - <body>
    - <items>
      - <item>
        <deptCd>9770703</deptCd>
        <empNm>강기운</empNm>
        <engNm>KANG Gi Yun</engNm>
        <hjNm>姜起潤</hjNm>
        <jpgLink>http://www.assembly.go.kr/photo/9770703.jpg</jpgLink>
        <num>2681</num>
        <origNm>경남 창원시청산구</origNm>
        <reeleGbnNm>초선</reeleGbnNm>
      </item>
      - <item>
        <deptCd>9770275</deptCd>
        <empNm>강기정</empNm>
        <engNm>KANG Gi Jung</engNm>
        <hjNm>姜琪正</hjNm>
        <jpgLink>http://www.assembly.go.kr/photo/9770275.jpg</jpgLink>
        <num>76</num>
        <origNm>광주 북구갑</origNm>
        <reeleGbnNm>3선</reeleGbnNm>
      </item>
      - <item>
        <deptCd>9770276</deptCd>
```