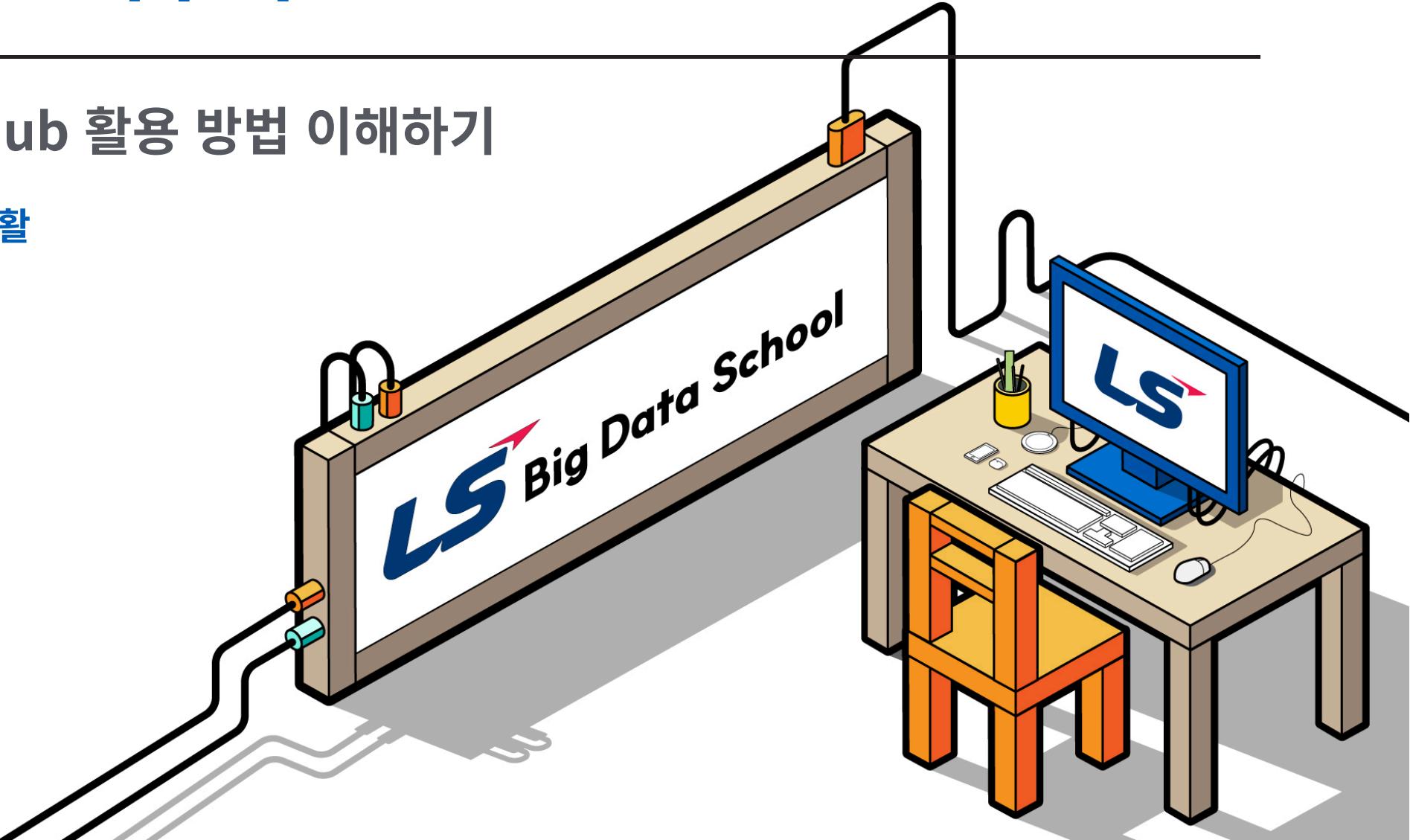


# Git 기초 배우기

---

Git과 Github 활용 방법 이해하기

슬기로운통계생활



# 코스 훑어보기.

Github의 기초 개념과 Github을 활용한 협업 방법에 대해 학습합니다.



# git 소개

Git은 **분산 버전 관리 시스템(DVCS)**입니다. 소스 코드 변경 이력을 관리하고, 여러 개발자가 동시에 작업할 수 있도록 지원합니다.

## 주요 기능

- 버전 관리: 코드 변경 내역을 기록하고 이전 상태로 복원할 수 있습니다.
- 분산 저장소: 모든 사용자가 로컬 저장소를 가지며, 중앙 서버 없이도 작업 가능합니다.
- 브랜치(branch): 독립적인 작업 공간을 생성하여 새로운 기능 개발이나 버그 수정 작업을 병렬로 진행할 수 있습니다.
- 병합(merge): 브랜치에서 작업한 내용을 다른 브랜치와 통합할 수 있습니다.
- 충돌 해결: 여러 사용자가 같은 파일을 수정했을 때 충돌(conflict)을 감지하고 해결할 수 있습니다.



# 버전 관리의 중요성



<https://www.youtube.com/watch?v=dMXq5eGWSf4>



# 버전 관리의 중요성

버전 관리는 파일의 변경 이력을 기록하고, 필요할 때 이전 상태로 되돌리거나 팀원과 작업을 병합 할 수 있는 시스템입니다.

## 버전 관리 예시

- 프로젝트 시작: 기본 파일 생성
  - 프로젝트 팀원인 Alice와 Bob은 팀 보고서를 작성하기 위해 report.docx 파일을 생성합니다.
  - 이 시점에서 파일의 상태는 아래와 같습니다.

[버전 1: Alice 작성]

"팀 프로젝트 보고서"



## 버전 관리 예시

- Alice가 내용을 추가
  - Alice는 팀 보고서에 첫 번째 섹션을 추가합니다.
  - Git을 사용하여 변경 사항을 기록(commit)합니다.

[버전 2: Alice 작성]

"팀 프로젝트 보고서

1. 프로젝트 개요: 이 프로젝트는..."

## 버전 관리 예시



- Bob이 내용을 추가
  - Bob은 두 번째 섹션을 추가합니다.
  - Bob 역시 Git을 사용하여 변경 사항을 기록(commit)합니다.

[버전 3 : Bob 작성]

"팀 프로젝트 보고서

1. 프로젝트 개요: 이 프로젝트는...
2. 데이터 분석 결과: 데이터는 다음과 같습니다..."

## 버전 관리 예시



- 충돌 상황

- Alice는 세 번째 섹션을 추가하고 내용을 약간 수정합니다.
- Bob은 같은 파일을 수정하여 제목을 변경하고, 두 번째 섹션을 업데이트합니다.

[버전 4: Alice 작성]

"팀 프로젝트 보고서

1. 프로젝트 개요: 이 프로젝트는...
2. 데이터 분석 결과: 데이터는 다음과 같습니다...
3. 결론: 분석 결과는..."

[버전 4: Bob 작성]

"프로젝트 팀 보고서 (업데이트됨)

1. 프로젝트 개요: 이 프로젝트는...
2. 데이터 분석 결과: 데이터는 통계적으로 유의미합니다."

## 버전 관리 예시



- Git을 사용한 충돌 해결
  - Alice와 Bob이 각각 git push 명령으로 원격 저장소에 변경 사항을 올리려 합니다.
  - Git은 두 사람이 같은 파일을 수정했으므로 **충돌(conflict)**이 발생했음을 알립니다.
  - 두 사람은 변경 사항을 비교하고, 필요한 내용을 병합합니다.

[버전 5: 병합 완료]

"프로젝트 팀 보고서 (업데이트됨)

1. 프로젝트 개요: 이 프로젝트는...
2. 데이터 분석 결과: 데이터는 통계적으로 유의미합니다.
3. 결론: 분석 결과는..."

병합된 내용을 git commit으로 기록하여 새로운 버전(버전 5)을 생성합니다.



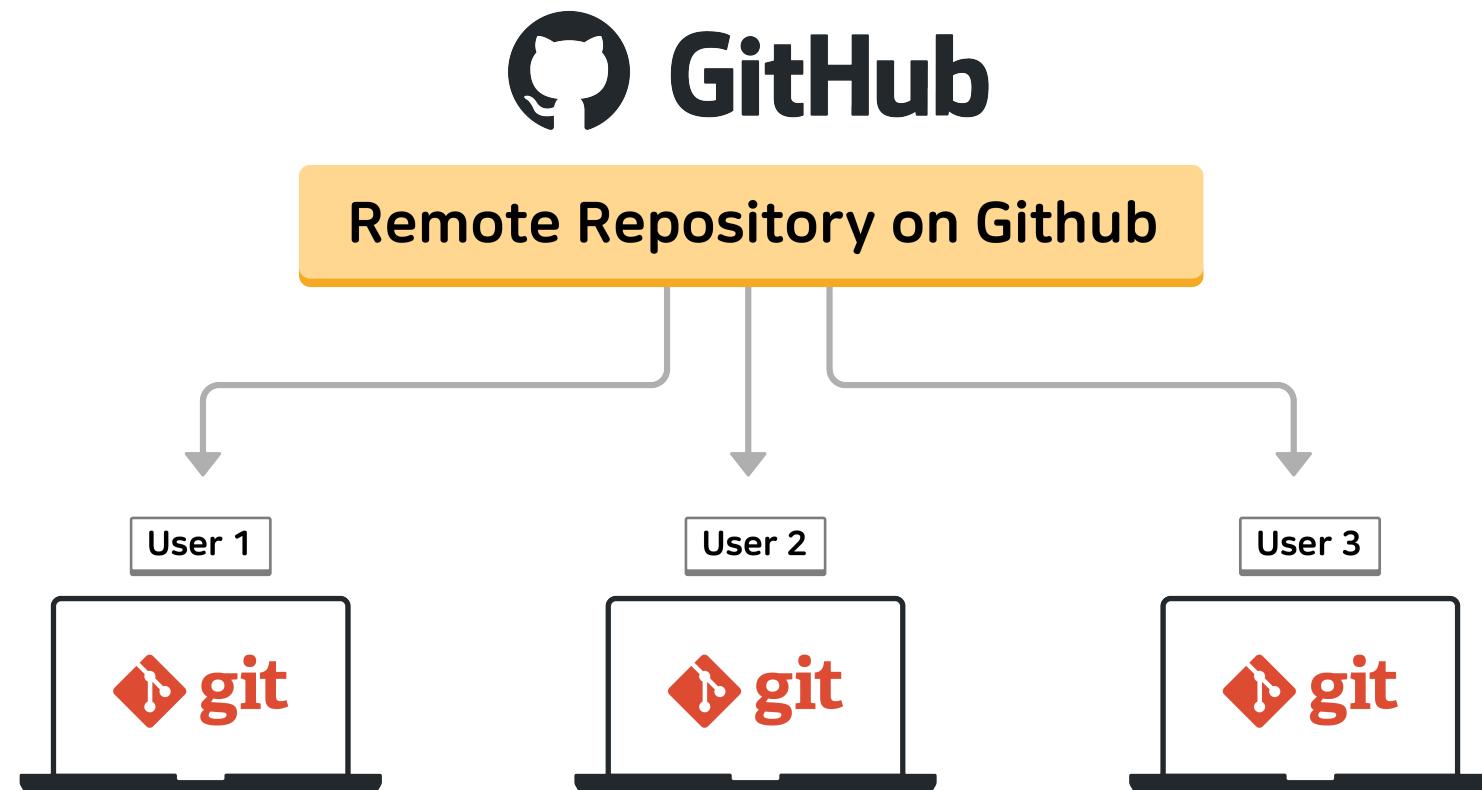
# 버전 관리 장점

- 이력 추적: 이전 버전으로 언제든지 되돌아갈 수 있습니다.
  - 예: 팀에서 실수로 내용을 삭제해도 버전 4로 복원 가능.
- 협업 용이: 여러 사람이 동시에 작업할 때도 충돌을 효율적으로 해결.
- 변경 기록 유지: 누가, 언제, 무엇을 변경했는지 명확히 알 수 있음.
- 백업 기능: 원격 저장소(GitHub 등)에 저장하여 파일 손실 방지.



# github 소개

GitHub는 Git을 활용해 협업을 돋는 원격 저장소 서비스로, 소프트웨어 개발자들이 코드를 저장, 관리, 공유하고 협업할 수 있도록 돋는 플랫폼입니다.





# github 소개

## 주요 기능

1. 원격 저장소: 프로젝트 코드를 클라우드에 저장하고 공유 가능.
2. 협업 도구: 코드 리뷰, 이슈 관리, Pull Request로 효율적인 팀워크 지원.
3. 버전 관리: Git과 통합되어 코드 변경 이력 기록 및 복원 가능.
4. CI/CD 자동화: GitHub Actions로 빌드, 테스트, 배포 워크플로우 자동화.
5. 프로젝트 관리: Kanban 보드, 마일스톤 등으로 작업 체계화.
6. 오픈소스 허브: 전 세계 개발자와 코드 공유 및 협업 가능.
7. GitHub Pages: 정적 웹사이트 무료 호스팅 기능 제공.
8. 보안 관리: 의존성 검사 및 Private Repository로 보안 강화.



# git 설치

윈도우 운영 체제에 맞게 설치하면 됩니다.

**주의 : 32비트/64비트 확인 후에 해당 비트에 맞는 프로그램을 설치해야 합니다.**

- git 설치 주소 : <https://git-scm.com/>

The screenshot shows the official Git website at https://git-scm.com/. The page features a large header with the Git logo and the tagline '--local-branching-on-the-cheap'. Below the header, there's a search bar and a brief introduction to what Git is and how it works. A central diagram illustrates the distributed nature of Git, showing multiple repositories connected by a network of arrows. The main content area includes sections for 'About', 'Documentation', 'Downloads', and 'Community'. On the right side, there's a section for the latest source release (2.47.1), download links for Mac, and links for Mac GUIs, Tarballs, Windows Build, and Source Code. At the bottom, there's a link to the book 'Pro Git' by Scott Chacon and Ben Straub.



# git 설치시 주의사항

PATH 환경 조정에 관한 설정이 나오면 **두 번째**를 클릭합니다. 나머지 페이지는 전부 기본값으로 세팅합니다.

## Adjusting your PATH environment

How would you like to use Git from the command line?



### Use Git from Git Bash only

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

### Git from the command line and also from 3rd-party software

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

### Use Git and optional Unix tools from the Command Prompt

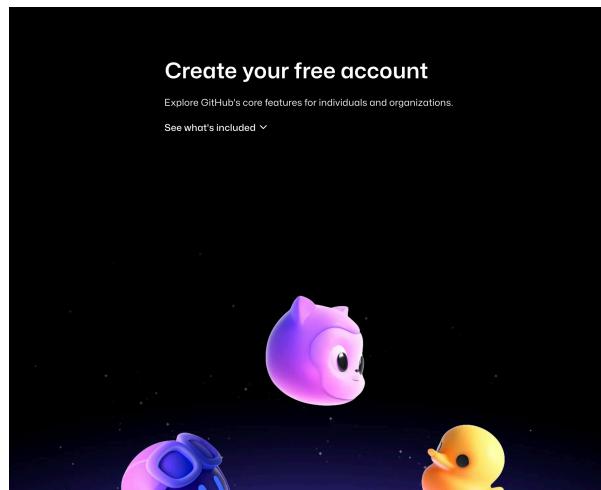
Both Git and the optional Unix tools will be added to your PATH.  
**Warning:** This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.



# github 회원가입

github 홈페이지에 접속하여 개인 계정을 생성합니다.

- github 주소 : <https://github.com/>



Already have an account? [Sign in ↗](#)

### Sign up to GitHub

Email<sup>\*</sup>

Password<sup>\*</sup>  
  
Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username<sup>\*</sup>  
  
Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

[Continue >](#)

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.



# git과 github 실습하기

- window -> git bash 창을 열어서 아래 코드를 실행합니다.

## git 설치 버전 확인

설치한 버전이 맞는지 확인합니다.

```
git --version
```

## git 계정 정보 등록

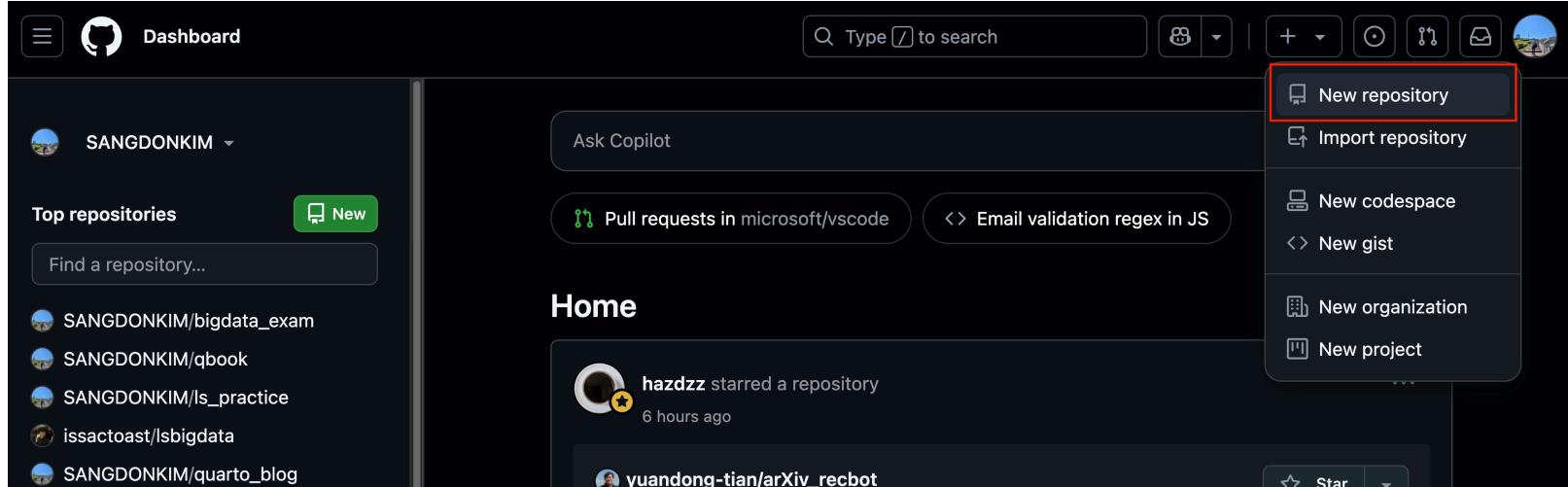
로컬에 설치한 git에 github 계정 정보를 등록합니다.

```
git config --global user.name "SANGDONKIM"  
git config --global user.email "your.email@example.com"
```

# github 레포지토리 생성



<https://github.com/> 페이지에서 새 레포지토리를 생성합니다.



The screenshot shows the GitHub Dashboard. On the left, there's a sidebar with the user's profile picture and name 'SANGDONKIM'. Below it, under 'Top repositories', are five listed repositories: 'bigdata\_exam', 'qbook', 'ls\_practice', 'lsbigdata', and 'quarto\_blog'. A green 'New' button is visible next to the repository list. In the center, there's a search bar with placeholder text 'Type ⌘ to search' and a dropdown menu with icons. To the right of the search bar is a '+' icon with a dropdown menu. This dropdown menu is open and contains several options: 'New repository' (which is highlighted with a red box), 'Import repository', 'New codespace', 'New gist', 'New organization', and 'New project'. Below the '+' icon, there are two cards: one for 'Pull requests in microsoft/vscode' and another for 'Email validation regex in JS'. At the bottom of the dashboard, there's a 'Home' section with a recent activity feed: 'hazdzz starred a repository 6 hours ago' and a link to 'yuandong-tian/arXiv\_recbot'. There's also a 'Star' button for this repository.

# github 레포지토리 생성



레포지토리 명을 지정한 후 저장소는 Public으로 설정합니다. ADD a README file을 클릭합니다.

**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 (\*).

**Repository template**

No template ▾

Start your repository with a template repository's contents.

**Owner \*** SANGDONKIM / **Repository name \*** myfirst\_repo myfirst\_repo is available.

Great repository names are short and memorable. Need inspiration? How about [bookish-octo-funicular](#) ?

**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](#).

## github 레포지토리 생성



myfirst\_repo라는 레포지토리가 생성되었고, 내부에는 README.md 파일이 생성된 것을 확인할 수 있습니다.

The screenshot shows the GitHub repository page for 'myfirst\_repo'. At the top, the repository name 'myfirst\_repo' is displayed under the user 'SANGDONKIM'. The 'Code' tab is selected. Below the header, there's a summary showing 'main' branch, '1 Branch', '0 Tags', and a commit from 'SANGDONKIM' labeled 'Initial commit' dated 'now'. A file named 'README.md' is listed with the same details. On the right side, there's an 'About' section with the message 'No description, website, or topics provided.' It also lists 'Readme', 'Activity', '0 stars', '1 watching', and '0 forks'. The 'Releases' section indicates 'No releases published' and a link to 'Create a new release'.



## github 레포지토리 연결

- 바탕화면에 github 레포지토리에 올리기 위한 로컬 파일을 생성합니다.
  - 파일명 : ls\_test
- 다음으로 터미널을 통해 해당 파일에 접근하기 위해 현재 경로를 확인합니다.

```
cd
```

cd는 Change Directory의 약자로, 현재 작업 중인 디렉토리(폴더)를 변경하는 명령어입니다.

- cd 명령어를 통해 ls\_test 파일에 접근합니다.

```
cd Desktop/ls_test
```

## github 레포지토리 연결



- git 초기화

```
git init
```

git init은 현재 디렉토리를 새로운 Git 저장소로 초기화하는 명령어입니다. 이 명령을 실행하면 Git이 해당 디렉토리를 버전 관리 대상으로 설정하고, Git 명령어를 사용할 수 있는 환경을 만듭니다.

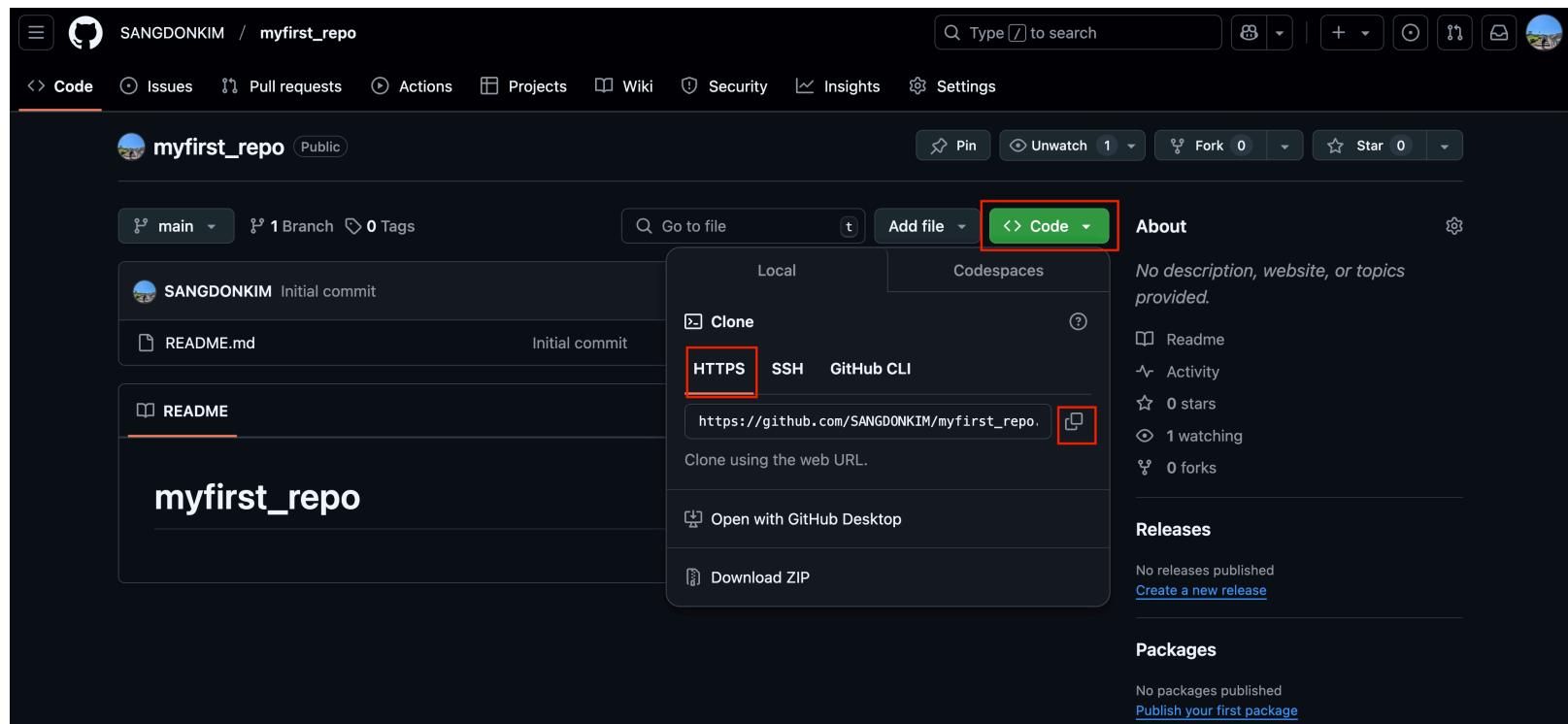
```
(base) sangdon@gimsangdon-ui-MacBookPro ~ % cd Desktop/ls_test
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git init
Initialized empty Git repository in /Users/sangdon/Desktop/ls_test/.git/
(base) sangdon@gimsangdon-ui-MacBookPro ls_test %
```



## github 레포지토리 연결

github 창에서 생성한 레포지토리의 url을 복사합니다.

- code -> HTTPS --> 복사 버튼 클릭



## github 레포지토리 연결



터미널 창에서 아래 코드를 입력합니다.

```
git remote add origin <복사한 url>
```

git remote add origin 명령은 로컬 Git 저장소를 원격 저장소(remote repository)와 연결하는 데 사용됩니다.

- remote: 원격 저장소를 관리하는 Git의 기능.
- add: 새 원격 저장소를 추가.
- origin: 원격 저장소의 기본 이름(사용자 정의 가능).
- URL: 연결하려는 원격 저장소의 주소.



## github 레포지토리 연결

연결이 잘되었는지 확인하기 위해 아래 코드를 입력합니다. fetch와 push 모두 지정한 url로 설정되어 있어야 합니다.

```
git remote -v
```

- fetch : 가져오기 작업에 활용되는 url
- push : 업로드 작업에 활용되는 url

```
(base) sangdon@gimsangdon-ui-MacBookPro ~ % cd Desktop/ls_test
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git init
Initialized empty Git repository in /Users/sangdon/Desktop/ls_test/.git/
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git remote add origin https://
github.com/SANGDONKIM/myfirst_repo.git
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git remote -v
origin https://github.com/SANGDONKIM/myfirst_repo.git (fetch)
origin https://github.com/SANGDONKIM/myfirst_repo.git (push)
(base) sangdon@gimsangdon-ui-MacBookPro ls_test %
```

## 원격 저장소에 로컬 저장소 파일 업로드하기



### git pull

원격 저장소와 로컬 저장소는 항상 최신화된 상태를 유지해야 합니다. 원격 저장소에 README.md 파일이 있으므로, 로컬 저장소에 파일을 가져오기 위해 git pull을 활용합니다.

```
git pull origin main
```

```
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 856 bytes | 214.00 KiB/s, done.
From https://github.com/SANGDONKIM/myfirst_repo
 * branch           main      -> FETCH_HEAD
 * [new branch]     main      -> origin/main
(base) sangdon@gimsangdon-ui-MacBookPro ls_test %
```

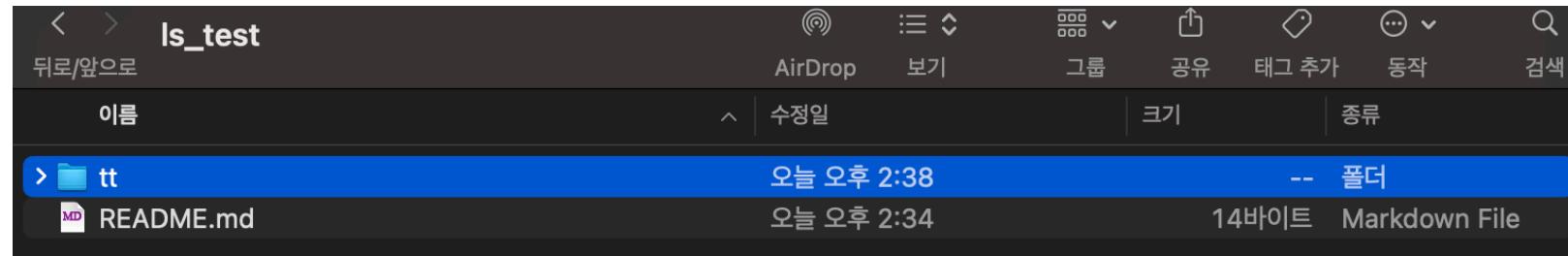
ls\_test 파일에 README.md 파일이 생성됩니다.



## 원격 저장소에 로컬 저장소 파일 업로드하기

로컬 저장소에 실습을 위해 파일을 생성합니다.

- 파일명 : tt
  - tt 파일 안에 .txt 파일 생성

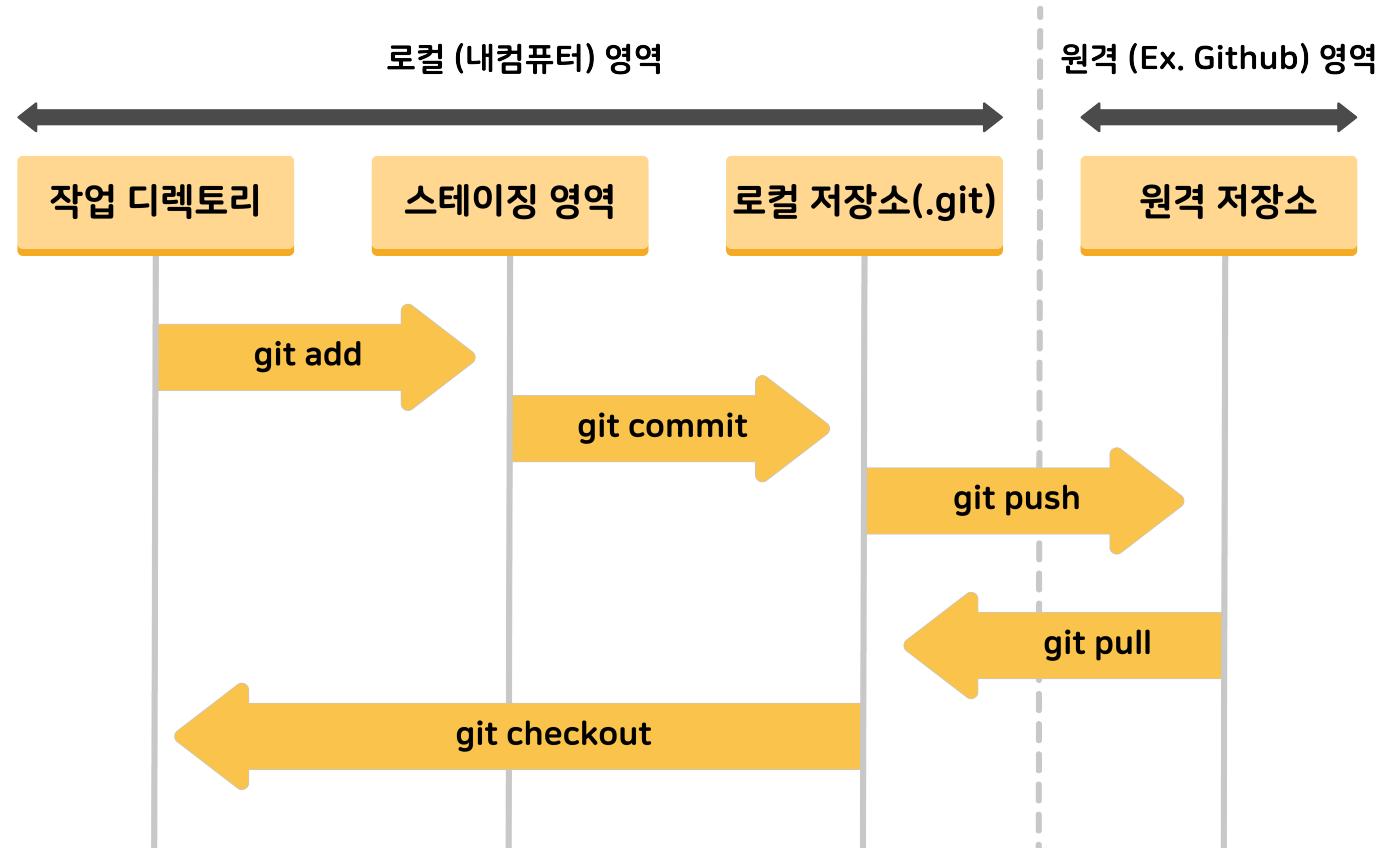


생성한 파일을 원격 저장소에 업로드해보겠습니다.



## 원격 저장소에 로컬 저장소 파일 업로드하기

원격 저장소에 로컬 저장소 파일을 업로드하기 위한 기본 절차는 다음과 같습니다. 여기서 스테이징 영역은 커밋할 변경 사항을 선택적으로 준비하기 위해 잠깐 거쳐가는 공간입니다.



# 원격 저장소에 로컬 저장소 파일 업로드하기



git add/commit/push 개요

요리책 만들기 예시

Add



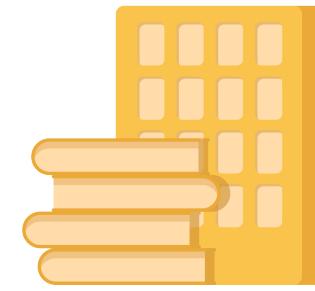
레시피 문서를 선택해서  
책에 넣을 준비를 할게!

Commit



이 레시피를 확정하고, 책에  
'새로운 디저트 레시피 추가'라는  
제목으로 저장했어!

Push



이 책을 출판사에 보냈어!  
이제 다른 팀원들도 볼 수 있어!

## 원격 저장소에 로컬 저장소 파일 업로드하기



### git add

git add는 Git에서 **변경된 파일을 스테이징 영역(Stage)**에 추가하는 명령어입니다. 이 명령어는 파일을 커밋(Commit) 준비 상태로 만들어 줍니다.

- `.` : 파일 전체를 의미함
- <파일경로/파일명> : 지정된 파일만 업로드

```
git add .
git add <파일경로/파일명>
```



## 원격 저장소에 로컬 저장소 파일 업로드하기

### git commit

git commit -m "커밋 메시지"는 **스테이징 영역(Stage)**에 있는 파일 변경 사항을 커밋(Commit)하고, 그 변경 사항에 대한 설명(메시지)을 추가하는 명령어입니다.

```
git commit -m "커밋 메시지"
```

```
(base) sangdon@gimsangdon-ui-MacBookPro ls_test % git commit -m 'update'  
[main 1fd2edc] update  
 1 file changed, 0 insertions(+), 0 deletions(-)  
  create mode 100644 .DS_Store
```

### 커밋 메시지의 중요성

- 커밋 메시지는 변경 사항을 다른 사람이나 자신이 이해할 수 있도록 설명하는 문서 역할을 합니다.
- 보통 변경사항에 대해 간결한 명령어로 보통 작성합니다.
- 예시 : git commit -m "Fix login button alignment issue"

## 원격 저장소에 로컬 저장소 파일 업로드하기



### git push

git push origin main은 **로컬 저장소(Local Repository)**에서 만든 변경 사항(커밋)을 **원격 저장소(Remote Repository)**에 업로드하는 명령어입니다.

```
git push origin main
```

```
Enumerating objects: 7, done.  
Counting objects: 100% (7/7), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (5/5), 469 bytes | 469.00 KiB/s, done.  
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
```

**warning** : 브랜치가 master로 되어있을 경우 main으로 변경 후 push 해야됨

```
git branch  
git branch -M main
```



## 원격 저장소에 로컬 저장소 파일 업로드하기

git push

github 원격 저장소를 확인해보면 tt 파일이 업로드된 것을 확인할 수 있습니다.

The screenshot shows a GitHub repository page for 'myfirst\_repo'. The 'Code' tab is selected. In the file list, there is a folder named 'tt' which is highlighted with a red box. Other files listed are '.DS\_Store' and 'README.md'. The commit history shows three commits: 'update' by SANGDONKIM (3 minutes ago), '.DS\_Store' (3 minutes ago), and 'Initial commit' (5 hours ago). The repository has 1 branch and 0 tags. On the right side, there is an 'About' section with a note: 'No description, website, or topics provided.' It also shows statistics: 3 commits, 0 stars, 1 watching, and 0 forks. The 'Releases' and 'Packages' sections are empty.

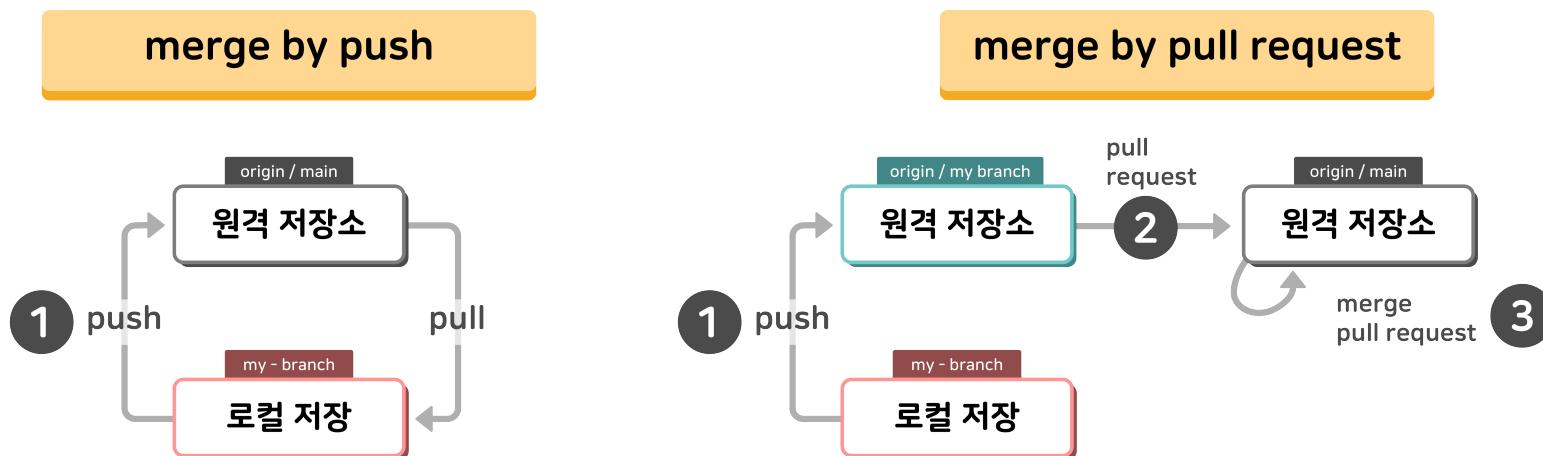


## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기

이전과 반대로 타인이 생성했던 원격 저장소 파일을 복사하여 로컬 저장소로 불러오겠습니다.

### Fork

**Fork(포크)**는 다른 사용자의 원격 저장소를 복사하여 나만의 원격 저장소로 가져오는 작업을 말합니다. Fork는 오픈소스 프로젝트에 기여하거나 기존 프로젝트를 기반으로 새로운 프로젝트를 개발할 때 사용됩니다.





## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기

### Fork

포크하고 싶은 저장소를 선택합니다.

저장소 링크 : [https://github.com/ddkim94/LS\\_fork\\_example](https://github.com/ddkim94/LS_fork_example)

The screenshot shows a GitHub repository page for 'LS\_fork\_example'. At the top, there's a navigation bar with tabs for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. Below the navigation bar, the repository name 'LS\_fork\_example' is displayed, along with its status as 'Public'. To the right of the repository name are buttons for Watch (1), Fork (0), and Star (0). A red box highlights the 'Fork' button. Below this, there's a summary card showing 'main' branch, 1 Branch, 0 Tags, and a recent commit by ddkim94 that updated README.md. On the right side of the page, there's an 'About' section which is currently empty, followed by sections for Readme, Activity, and Statistics (0 stars, 1 watching, 0 forks). At the bottom, there are sections for Releases and Report repository.



# 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기

## Fork

설정 변경 없이 Create fork를 클릭합니다.

The screenshot shows the GitHub interface for creating a new fork of a repository named 'ddkim94/LS\_fork\_example'. The 'Code' tab is selected in the navigation bar. The main area is titled 'Create a new fork' and explains that a fork is a copy of a repository. It shows the owner as 'ddkim94' and the repository name as 'LS\_fork\_example'. A note indicates that the name is available. Below this, there's a 'Description (optional)' input field, a checked checkbox for 'Copy the main branch only', and a note about contributing back. At the bottom right is a green 'Create fork' button with a red border.



## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기

### git clone

Fork된 저장소를 로컬 환경으로 복제합니다.

The screenshot shows a GitHub repository page for 'LS\_fork\_example'. The 'Code' dropdown menu is open, and the 'Clone' option is highlighted with a red box. The URL 'https://github.com/SANGDONKIM/LS\_fork\_exam' is visible in the 'Clone' section. The repository has 1 branch and 0 tags. The README file contains the text 'This branch is up to date with ddkim94/LS\_fork\_example:main.'

Code

LS\_fork\_example (Public)

forked from ddkim94/LS\_fork\_example

main 1 Branch 0 Tags

This branch is up to date with ddkim94/LS\_fork\_example:main.

ddkim94 Update README.md

README.md Update README.m

README

LS\_fork\_example

이름 : SANGDONKIM

Pin Watch 0 Fork 0 Star 0

Code

Local Codespaces

Clone

HTTPS SSH GitHub CLI

https://github.com/SANGDONKIM/LS\_fork\_exam

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기



### git clone

저장하고 싶은 위치로 이동합니다.

```
cd ~/Desktop/폴더이름
```

Fork된 저장소를 로컬 환경으로 복제합니다.

```
git clone https://github.com/SANGDONKIM/LS_fork_example.git
```

```
(base) sangdon@gimsangdon-ui-MacBookPro Desktop % git clone https://github.com/SANGDONKIM/LS_fork_example.git
Cloning into 'LS_fork_example'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기



**git clone**

LS\_fork\_example 파일로 이동합니다.

```
cd LS_fork_example
```

현재 로컬 Git 저장소에 설정된 원격 저장소(remote repository) 목록과 관련 URL을 확인합니다.

```
git remote -v
```

```
(base) sangdon@gimsangdon-ui-MacBookPro LS_fork_example % git remote -v
origin  https://github.com/SANGDONKIM/LS_fork_example.git (fetch)
origin  https://github.com/SANGDONKIM/LS_fork_example.git (push)
```

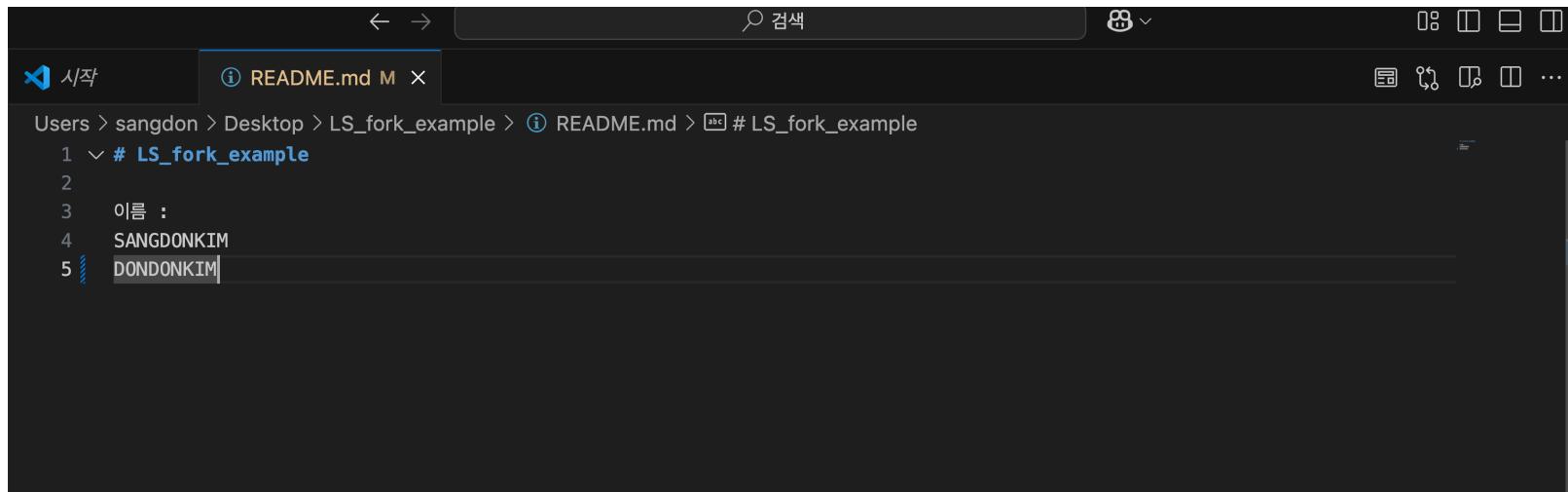
fetch/push 모두 복제한 원격 저장소 url이어야 합니다.



## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기

### git clone

LS\_fork\_example 폴더 안 README.md 파일을 VScode로 열고, 파일에 본인 이름을 추가하고 저장합니다.



```
Users > sangdon > Desktop > LS_fork_example > README.md > # LS_fork_example
1 # LS_fork_example
2
3 이름 :
4 SANGDONKIM
5 DONDONKIM
```

## 타인이 생성한 원격 저장소 파일 로컬 저장소로 가져오기



### 수정된 파일을 원격 저장소에 업로드하기

이전처럼 add/commit/push를 활용해서 업데이트합니다.

```
git add .
```

```
git commit -m 'add name dondonkim'
```

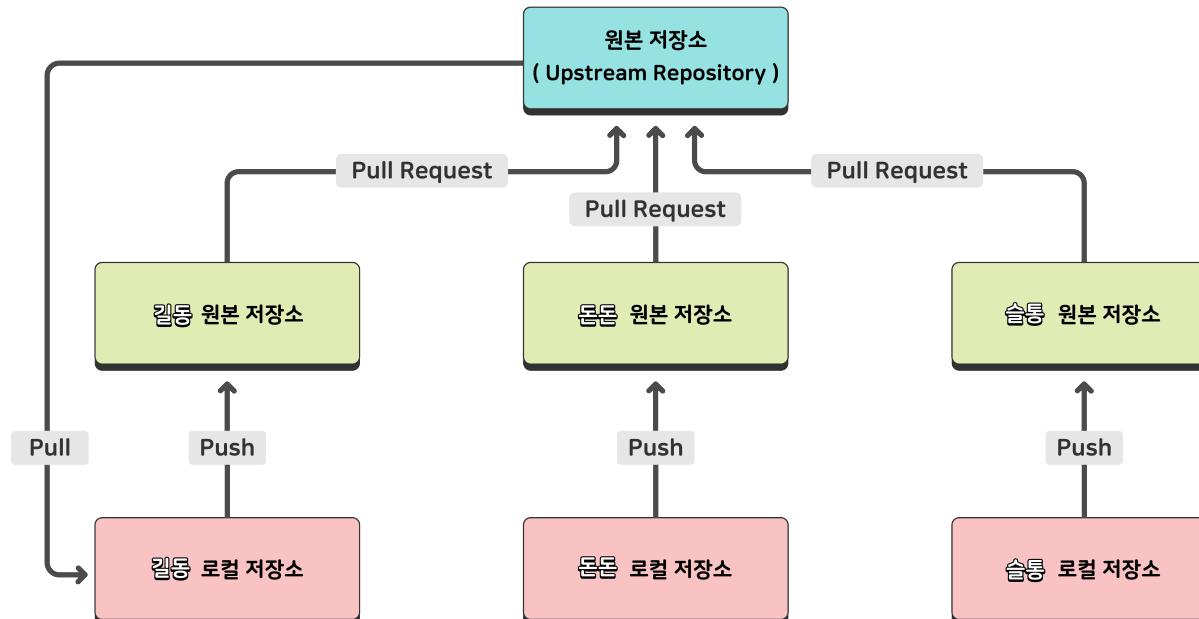
```
git push origin main
```



## 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

git push를 통해 원격 저장소에 업데이트했지만, 해당 원격 저장소는 원본 원격 저장소가 아닌 복제된 원격 저장소입니다. 만약 원본 원격 저장소에 변경된 내용을 업로드하고 싶다면 pull/request를 활용해야 합니다.

### pull/request



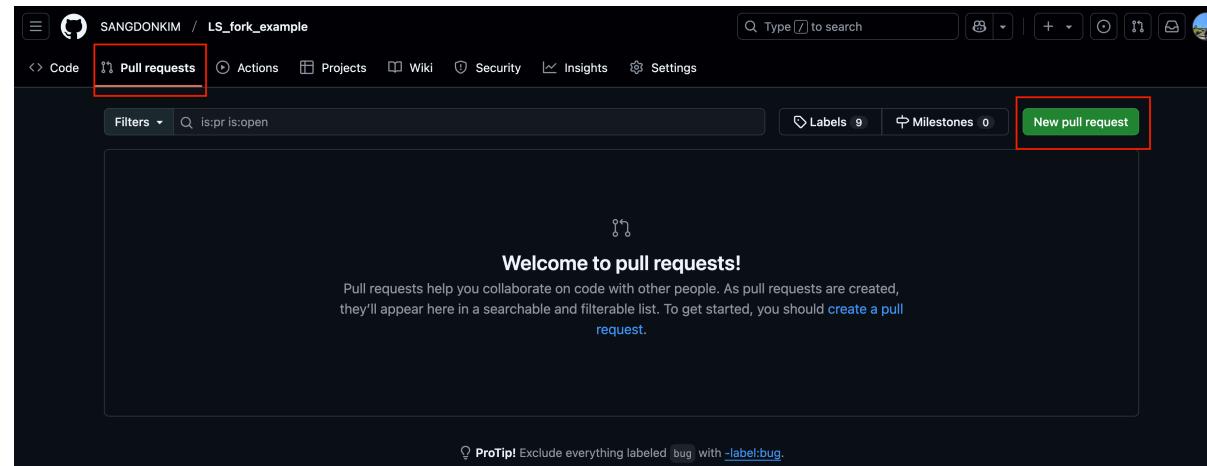


## 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

### Pull Request(PR)

**Pull Request**는 GitHub, GitLab 등의 협업 플랫폼에서 사용하는 기능으로, 원본 저장소에 변경 사항을 병합(Merge)해달라고 요청하는 과정입니다.

- fork한 저장소 --> Pull requests --> New pull request





# 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

## Pull Request(PR)

- Create pull request 클릭

변경 사항이 기록된 것을 확인할 수 있습니다.

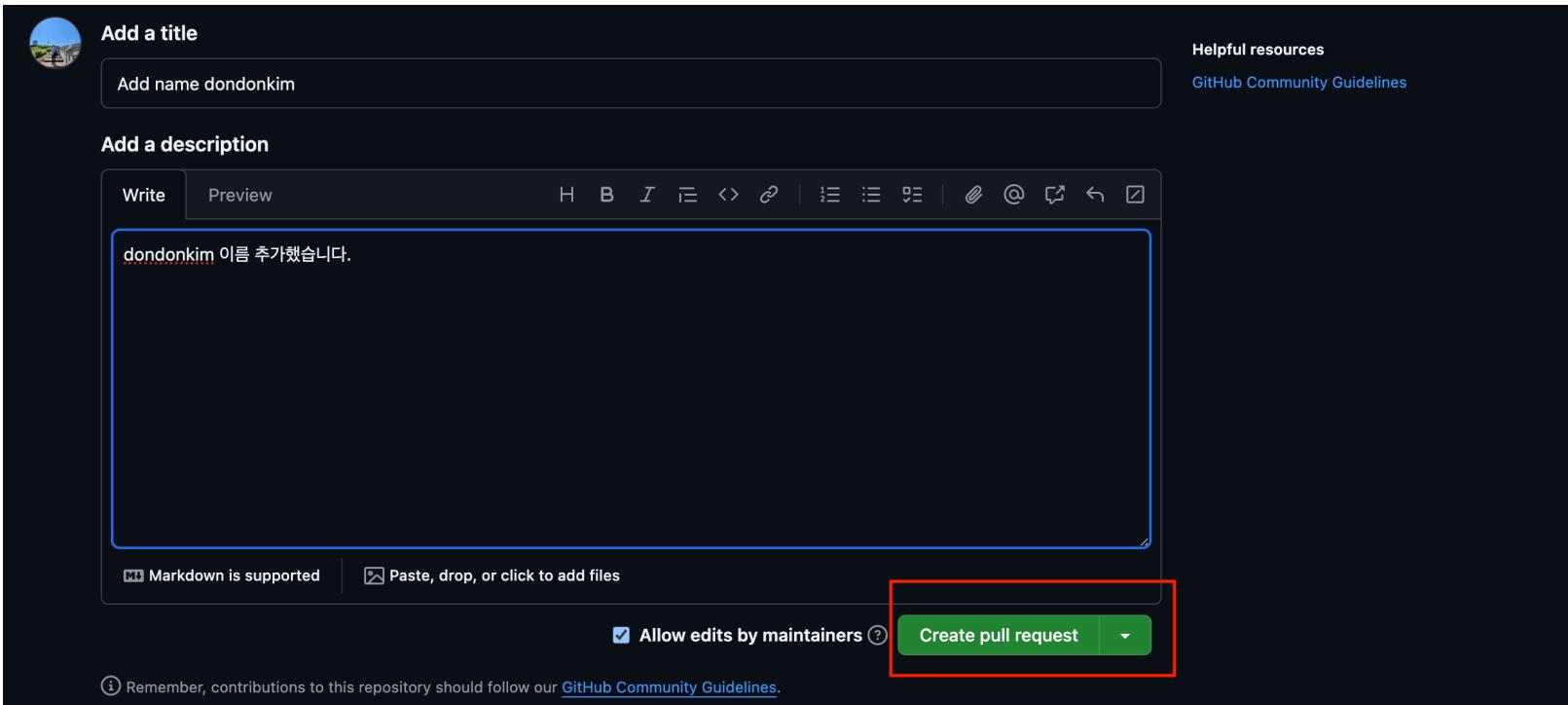
The screenshot shows a GitHub repository comparison interface. At the top, it says "Comparing changes" between "base repository: ddkim94/LS\_fork\_example" and "head repository: SANGDONKIM/LS\_fork\_exam...". It indicates that the branches are "Able to merge". Below this, there's a message about discussing changes and a prominent green "Create pull request" button, which is highlighted with a red box. The main area shows a commit from "dondonkim" on Jan 7, 2025, with one file changed and one contributor. The commit message includes Korean text: "이름 : SANGDONKIM" and "+ DONDONKIM". At the bottom, a diff view of the README.md file is shown, highlighting the addition of the new name.



## 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

### Pull Request(PR)

- Create pull request 클릭



The screenshot shows the GitHub interface for creating a pull request. At the top, there's a placeholder for 'Add a title' with a profile picture placeholder and a note to 'Add name dondonkim'. Below it is a 'Add a description' section with a 'Write' tab selected, showing the text 'dondonkim 이름 추가했습니다.' (Added dondonkim's name). To the right of the text area are various rich text editing icons. On the far right, under 'Helpful resources', are links to 'GitHub Community Guidelines' and 'GitHub Help'. At the bottom, there are two buttons: 'Markdown is supported' and 'Paste, drop, or click to add files'. A red box highlights the 'Create pull request' button, which is green with white text. To its left is a checkbox labeled 'Allow edits by maintainers' with a question mark icon. A small note at the bottom says 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)'.

# 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기



## Pull Request(PR) 메시지 양식

### ### 제목 (Title)

[변경 유형] 변경 내용을 간략히 요약

### ### 변경 내용 (Description)

- 변경 내용 요약 (무엇이 변경되었는지)
- 변경 이유 (왜 변경이 필요한지)
- 구현 방식 (어떻게 구현했는지)
- 테스트 방법 (변경 사항을 검증한 방법)

### ### 관련 이슈 (Related Issues)

- Fixes #123
- Closes #456

### ### 체크리스트 (Checklist)

- [x] 코드가 정상적으로 작동하는지 테스트 완료
- [x] 관련 문서 업데이트
- [x] 팀원의 리뷰 반영



## 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

### Pull Request(PR)

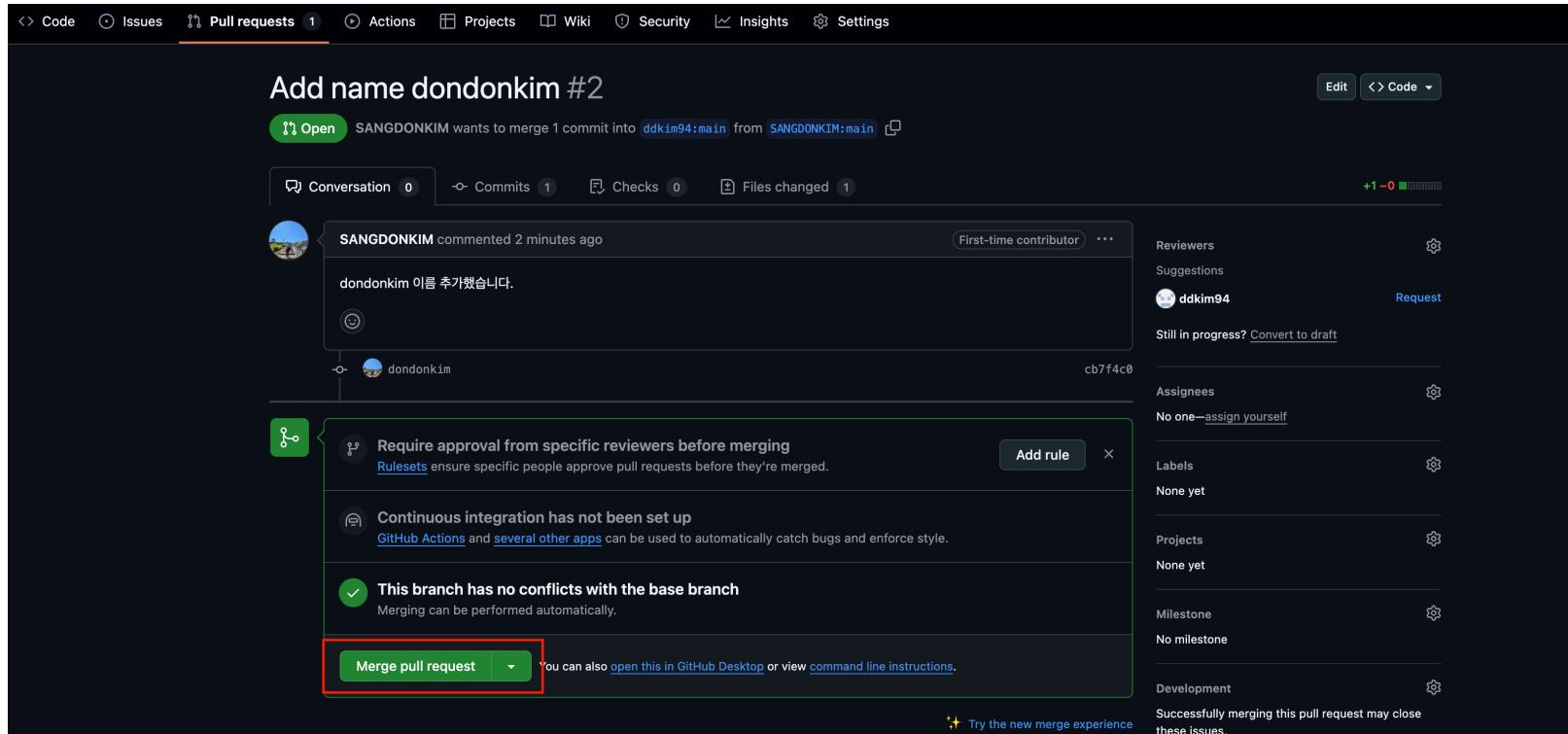
PR을 보냈으므로, 원본 저장소의 관리자가 검토 후 병합(merge) 여부를 결정합니다.

The screenshot shows a GitHub Pull Request page for a repository. The title is "Add name dondonkim #2". The status is "Open" and it shows a merge from "ddkim94:main" to "SANGDONKIM:main". There is 1 commit, 0 checks, and 1 file changed. A comment from "SANGDONKIM" says "dondonkim 이름 추가했습니다.". The status bar at the bottom says "This branch has no conflicts with the base branch". On the right side, there are sections for "Reviewers" (No reviews), "Assignees" (No one assigned), "Labels" (None yet), and "Projects" (None yet). A button "Edit" is visible at the top right.



## 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

관리자일 경우 다음과 같은 메시지가 나오며, **Merge pull request**를 클릭하면 최종 병합됩니다.



The screenshot shows a GitHub pull request interface for a repository named "Add name dondonkim #2". The pull request is from user "SANGDONKIM" to branch "ddkim94:main". The commit message is "dondonkim 이름 추가했습니다." (Added dondonkim name). The pull request has 1 commit, 0 checks, and 1 file changed. The status bar indicates "+1 -0" reviews.

On the right side, there are sections for "Reviewers" (ddkim94), "Suggestions" (Request), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (None yet).

A modal window titled "Merge pull request" is open at the bottom left. It contains three items:

- "Require approval from specific reviewers before merging" (with a "Rulesets" link)
- "Continuous integration has not been set up" (with a "GitHub Actions" and "several other apps" link)
- "This branch has no conflicts with the base branch" (with a "Merging can be performed automatically." note)

At the bottom of the modal, there is a green "Merge pull request" button with a dropdown arrow, which is highlighted with a red box. Below it, a note says "You can also open this in GitHub Desktop or view command line instructions." A small note at the bottom right says "Try the new merge experience".



# 타인이 생성한 원격 저장소와 포크한 저장소 동기화하기

## Pull Request(PR)

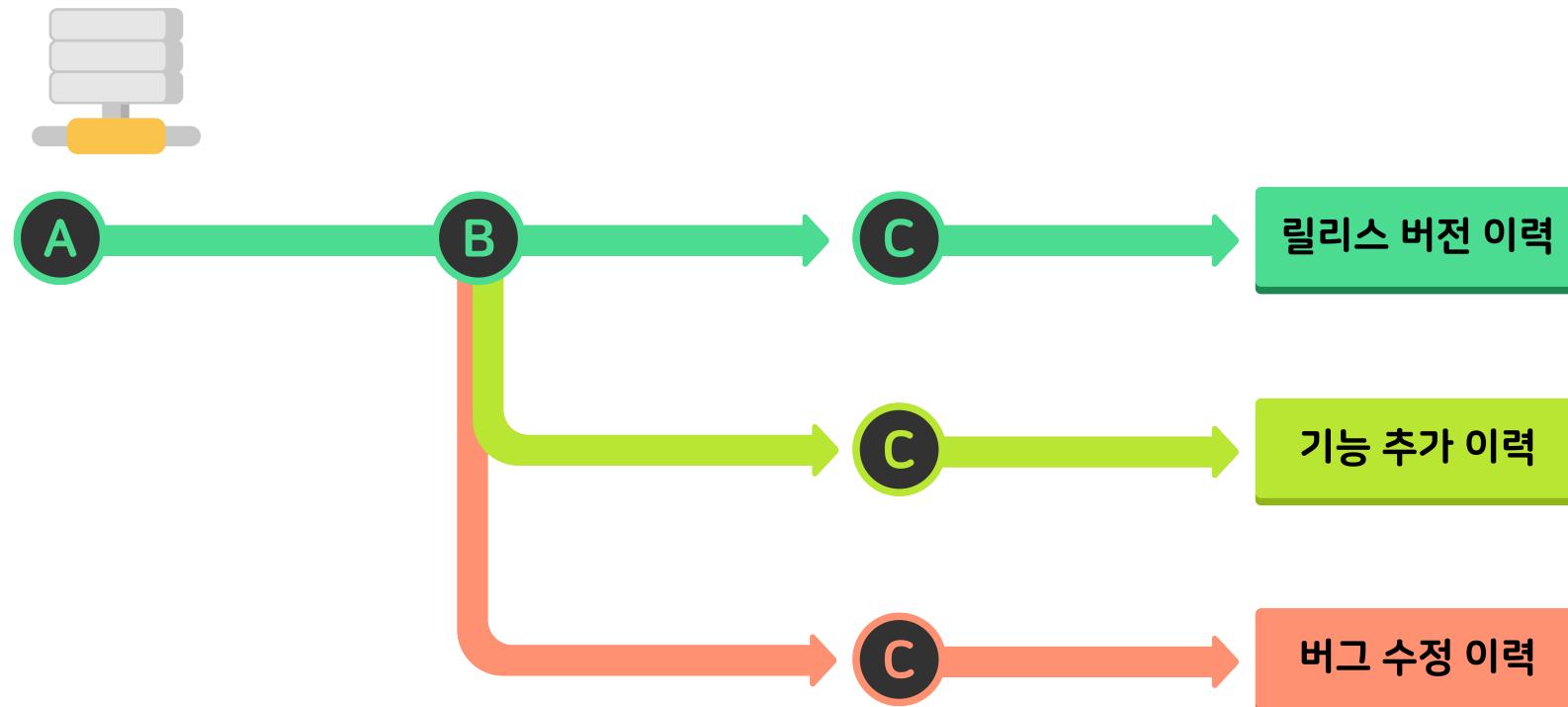
merge를 완료한 결과를 보면 원본 저장소에 dondonkim 이름이 추가된 것을 확인할 수 있습니다.

The screenshot shows a GitHub repository page for 'LS\_fork\_example'. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Code' tab is selected. In the center, there's a card for a merge pull request from 'ddkim94' to 'main'. The card includes the commit message: 'Merge pull request #2 from SANGDONKIM/main', the author 'dodonkim', and the timestamp '1 minute ago'. Below the card, there's a 'README' file by 'dodonkim' updated '1 hour ago'. On the right side, there's an 'About' section with the message 'No description, website, or topics provided.' It also lists 'Readme', 'Activity', '0 stars', '1 watching', and '1 fork'. At the bottom, there's a 'Releases' section with the message 'No releases published' and 'Create a new release'.



# 브랜치 활용하기

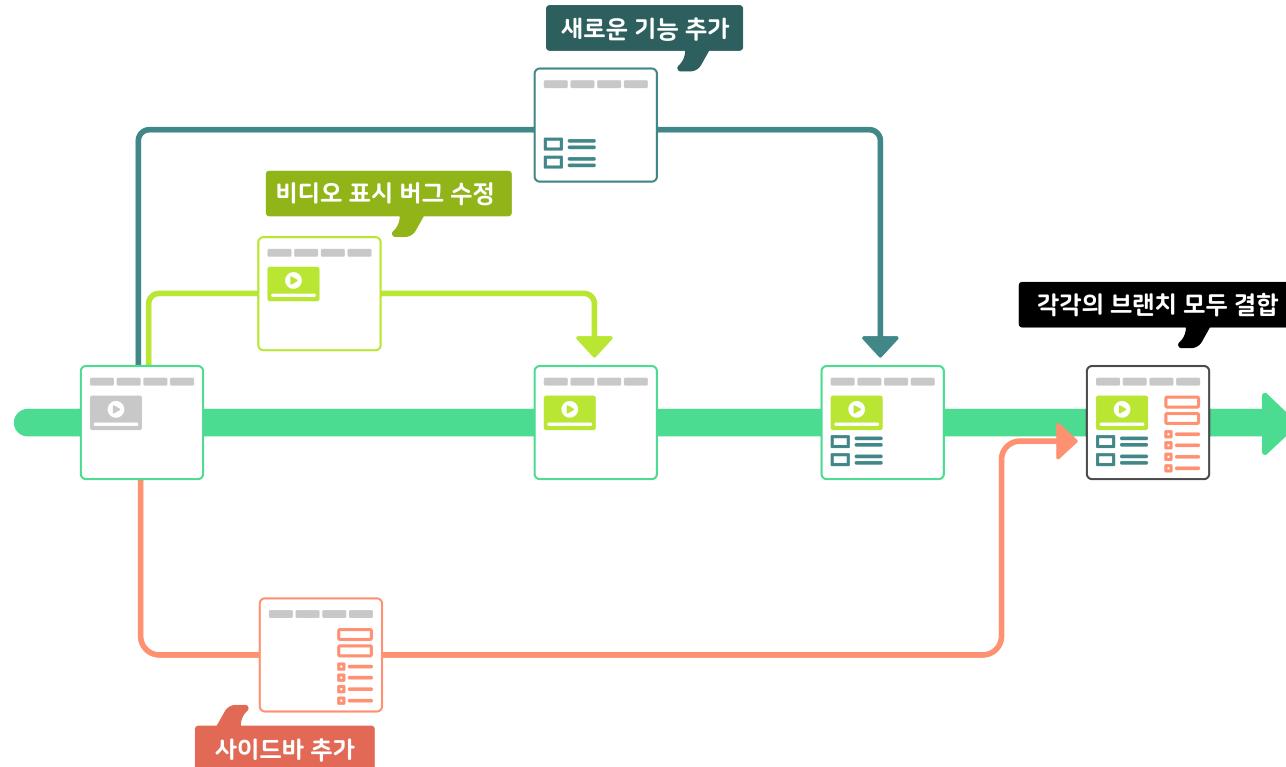
**브랜치(Branch)**는 Git에서 하나의 독립적인 작업 공간을 의미합니다. 브랜치를 사용하면 기존 코드(main 브랜치 등)를 그대로 유지하면서 새로운 기능 개발, 버그 수정, 실험 등을 안전하게 진행할 수 있습니다. 브랜치는 작업 완료 후 기본 브랜치에 병합(Merge)하거나 삭제할 수 있습니다.





# 브랜치 활용하기

각자의 브랜치를 생성한 후 다른 사람의 작업에 영향을 받지 않고 독립적으로 특정 작업을 수행하고 그 결과를 하나로 모아 나가게 됩니다. 브랜치로 작업의 기록을 남기게 되므로 문제가 발생했을 때 원인을 찾기 쉬워집니다.



## 브랜치 생성하기



firstbranch라는 브랜치를 생성합니다.

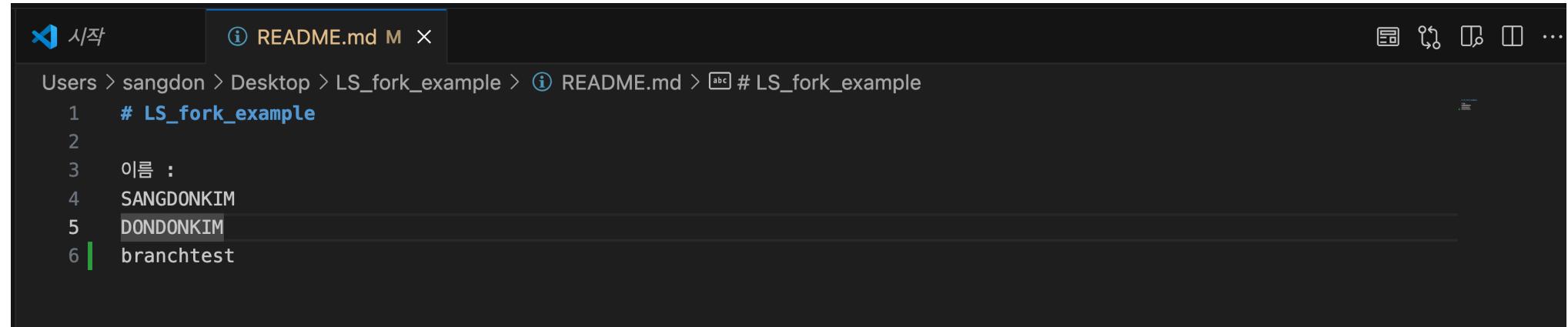
```
git checkout -b <branch-name>
```

```
(base) sangdon@gimsangdon-ui-MacBookPro LS_fork_example % git checkout -b firstbranch
Switched to a new branch 'firstbranch'
```



## 코드 변경하기

README.md 파일에 branchtest라는 텍스트를 추가했습니다.



```
 시작 README.md X
Users > sangdon > Desktop > LS_fork_example > README.md # LS_fork_example
1 # LS_fork_example
2
3 이름 :
4 SANGDONKIM
5 DONDONKIM
6 branchtest
```



## 코드 변경하기

수정된 파일을 add/commit 합니다.

```
git add .
git commit -m "add branchtest"
```

```
(base) sangdon@gimsangdon-ui-MacBookPro LS_fork_example % git add .
(base) sangdon@gimsangdon-ui-MacBookPro LS_fork_example % git commit -m 'add branchtest'
[firstbranch a545968] add branchtest
 1 file changed, 2 insertions(+), 1 deletion(-)
```



## 원격 저장소에 브랜치 푸시하기

firstbranch 브랜치에 수정된 내용을 푸시합니다.

```
git push origin firstbranch
```

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'firstbranch' on GitHub by visiting:
remote:     https://github.com/SANGDONKIM/LS_fork_example/pull/new/firstbranch
remote:
To https://github.com/SANGDONKIM/LS_fork_example.git
 * [new branch]      firstbranch -> firstbranch
```

# P/R 보내기



firstbranch 브랜치에 수정된 내용을 P/R합니다.

The screenshot shows the GitHub interface for creating a pull request. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The 'Code' link is highlighted with an orange underline. Below the navigation, the title 'Comparing changes' is displayed, followed by a sub-instruction: 'Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#)'. A message indicates that the branches are 'Able to merge'. The base repository is set to 'ddkim94/LS\_fork\_example' and the base branch is 'main'. The head repository is 'SANGDONKIM/LS\_fork\_exam...' and the compare branch is 'firstbranch'. In the main area, there is a section to 'Add a title' with the placeholder 'add branchtest'. Below it, there is a section to 'Add a description' with a rich text editor interface showing 'Write' and 'Preview' tabs, and a text input field with the placeholder 'Add your description here...'. A note states 'Markdown is supported'. At the bottom, there is a checkbox for 'Allow edits by maintainers' and a green button labeled 'Create pull request'.

# P/R 보내기



충돌 없이 병합될 수 있다는 문구가 출력된 것을 확인할 수 있습니다.

add branchtest #3

[Open](#) SANGDONKIM wants to merge 1 commit into `ddkim94:main` from `SANGDONKIM:firstbranch` ⚙

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 -1

SANGDONKIM commented now  
No description provided.  
add branchtest a545968

This branch has no conflicts with the base branch  
Only those with [write access](#) to this repository can merge pull requests.

Contributor ...

Reviewers  
No reviews  
Still in progress? [Convert to draft](#)

Assignees  
No one assigned

Labels  
None yet

Projects

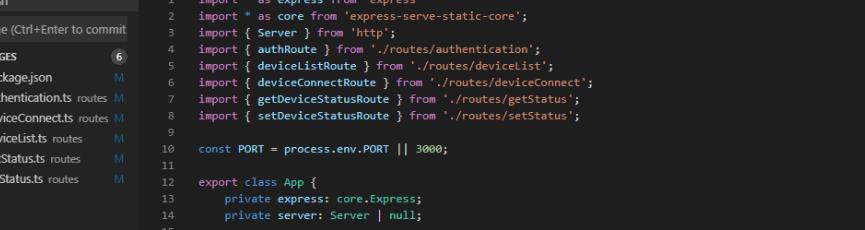


# VScode에서 github 활용하기

VScode에는 GUI 방식으로 쉽게 github을 연동할 수 있도록 지원합니다. Github 연동 및 활용을 위한 확장 프로그램 몇 가지를 소개하겠습니다.

# Git Graph

작업 중인 레포지토리에서 커밋 로그를 그래프 형태로 보여줍니다.



```
import * as express from 'express'
import * as core from 'express-serve-static-core';
import { Server } from 'http';
import { authRoute } from './routes/authentication';
import { deviceListRoute } from './routes/deviceList';
import { deviceConnectRoute } from './routes/deviceConnect';
import { getDeviceStatusRoute } from './routes/getStatus';
import { setDeviceStatusRoute } from './routes/setStatus';

const PORT = process.env.PORT || 3000;

export class App {
    private express: core.Express;
    private server: Server | null;

    constructor() {
        this.express = express();
        this.registerRoutes();
    }

    public start() {
        this.server = this.express.listen(PORT, () => {
            console.log('App running on port ' + PORT);
        });
    }

    public stop() {
        if (this.server) this.server.close(() => {
            console.log('App stopped');
        });
    }

    private registerRoutes(): void {
        ...
    }
}
```

## Git Lens



코드의 특정 단락에 해당 내용이 누가, 언제 작성한 것인지 커밋 정보를 흐린 글씨로 표시해 줍니다.

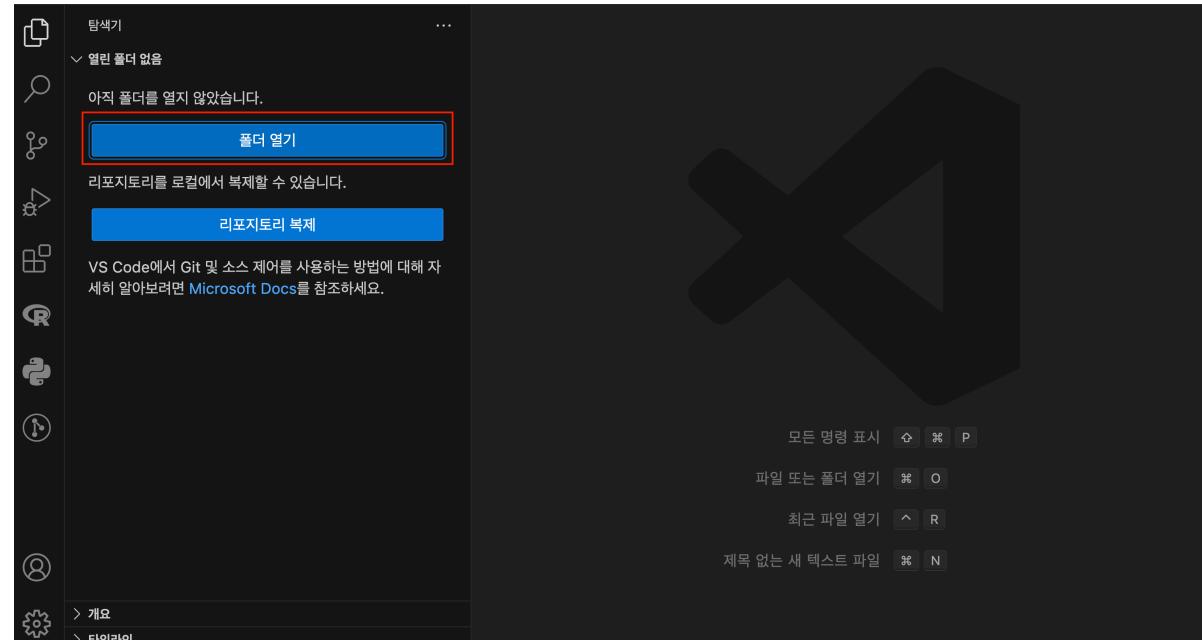
```
function gitLens(code: string) {
  return supercharged(code);    You, 4 years ago • Supercharged
}
```



# VScode에서 github 활용하기

이전에 실습했던 LS\_fork\_example 파일을 활용하겠습니다. 해당 파일은 git init을 통해 이미 git으로 연동이 되었으므로, VScode에서도 동일하게 적용됩니다.

- 폴더 열기 -> LS\_fork\_example 파일 열기





# VScode에서 github 활용하기

LS\_fork\_example 파일에 있는 README.md 파일에 VS CODE TEST라는 문구를 추가하겠습니다.

- README.md 파일 열기 -> VS CODE TEST 문구 추가 -> 저장

```
① README.md M X
① README.md > # LS_fork_example
You, 1초 전 | 2 authors (ddkim94 and one other)
1 # LS_fork_example
2
3 이름 :
4 SANGDONKIM
5 DONDONKIM
6 branchtest
7 VS CODE TEST You, 1초 전 • Uncommitted changes
```

저장을 완료하면 README.md에 M(Modified)라는 문구가 표시됩니다.



# VScode에서 github 활용하기

LS\_fork\_example 파일에 test.md 파일을 추가하겠습니다.

- 탐색기 -> +폴더 버튼 클릭 -> test.md 입력 -> test.md 파일 생성



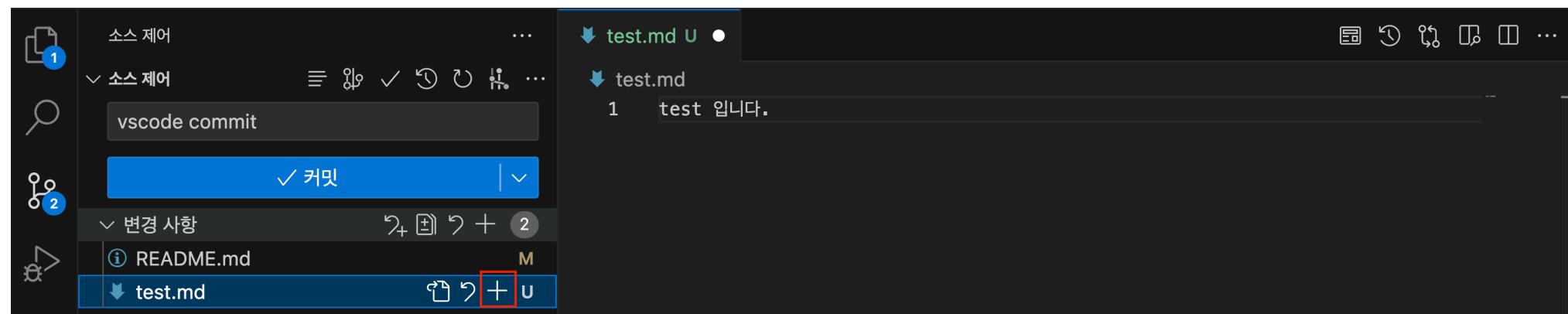


# VScode에서 github 활용하기

test.md 파일 저장을 완료하면 U(Untracked) 문구가 표시됩니다. 즉, 원격 저장소에 아직 추가되지 않은 파일이라는 의미입니다.

## git add

- add 하고 싶은 파일 옆 + 버튼 클릭
  - README.md, test.md 추가

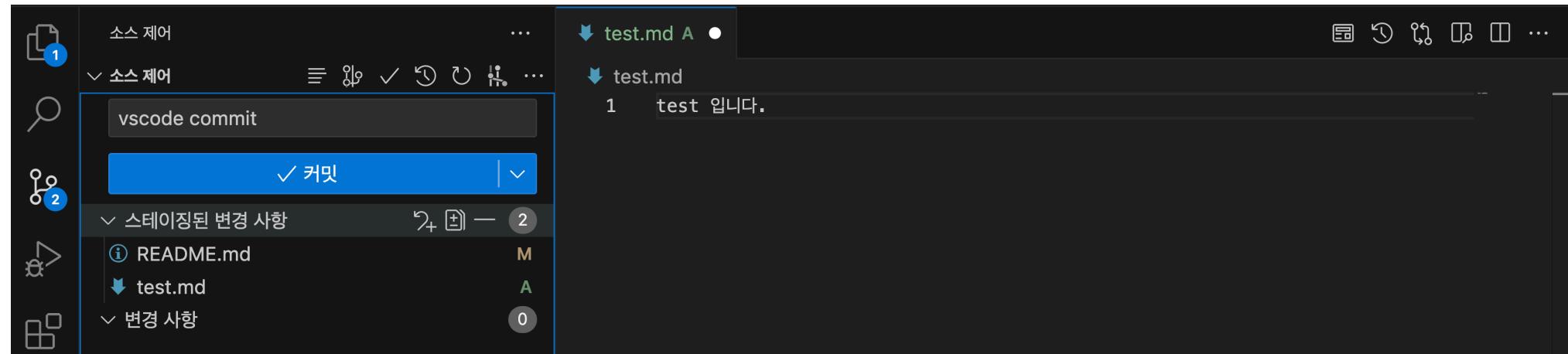




# VScode에서 github 활용하기

# git add

스테이징된 변경사항으로 2개 파일이 추가된 것을 확인할 수 있습니다.





# VScode에서 github 활용하기

## git commit

커밋 문구를 입력하고, 커밋을 진행합니다.

- vscode commit 문구 입력 -> 커밋 버튼 클릭

The screenshot shows the VS Code interface with the GitLens extension installed. On the left, the Explorer sidebar shows a folder named '소스 제어' containing files 'README.md' and 'test.md'. The status bar indicates there are 2 changes (M, A) and 0 staged changes (0). In the center, the GitLens interface displays the commit history for 'test.md'. The first commit is shown with the message '1 test 입니다.' and a status of 'A'. Below the commit list, a command palette is open, showing the 'vscode commit' command highlighted with a red box. To its right, a button labeled '✓ 커밋' (Commit) is also highlighted with a red box. The status bar at the bottom right shows the commit count '1'.

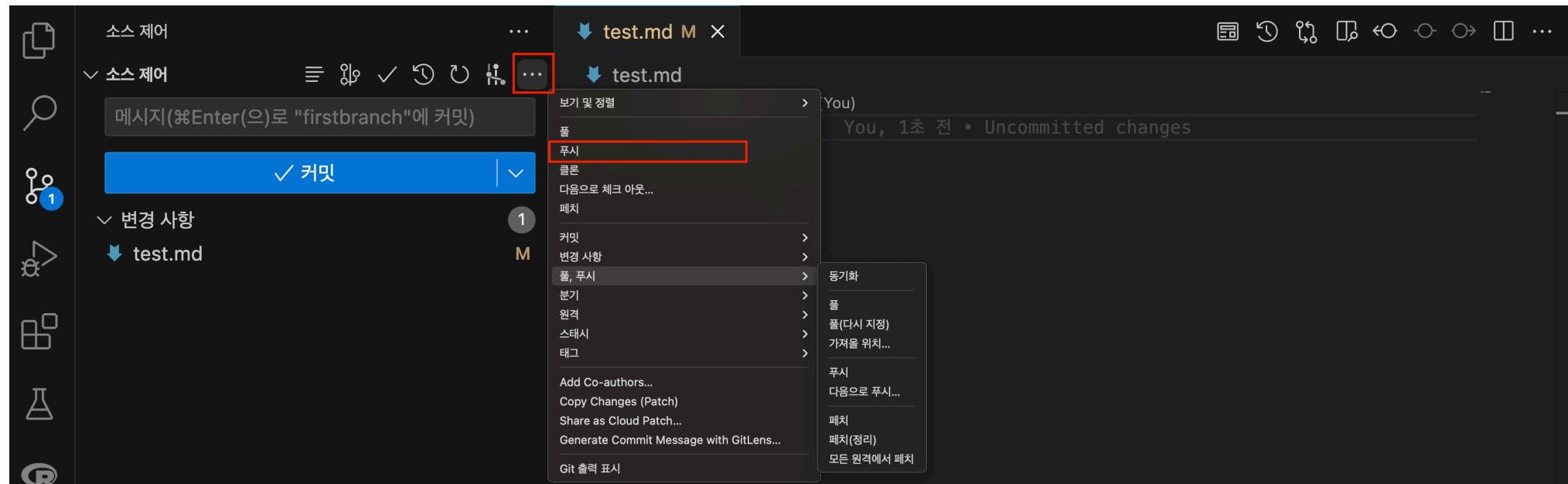


# VScode에서 github 활용하기

## git push

푸시를 진행합니다.

- ... 버튼 클릭 > 푸시 버튼 클릭





# git 주요 용어 정리

- git add .로 스테이징 영역에 추가된 결과를 취소하고 싶을 때

```
git reset
```

- git add .로 스테이징 영역에 추가된 일부 결과를 취소하고 싶을 때

```
git reset <파일경로/파일명>
```

- 커밋 메시지 수정하고 싶을 때

```
git commit --amend
```

- push 전 커밋을 취소하고 싶을 때

```
git reset --soft HEAD^
```



# git 주요 용어 정리

- Git에서 현재 작업 디렉토리의 상태를 확인하고 싶을 때

```
git status
```

- 커밋 히스토리를 확인하고 싶을 때

--oneline : 커밋 해시와 커밋 메시지만 출력

```
git log --oneline
```



# git 주요 용어 정리

- 커밋 내역 확인하고 싶을 때
  - 커밋 메시지에는 421b2f7와 같이 고유의 코드로 기록되며, 해당 코드로 조회 가능

```
git show 421b2f7
```

Commits

main ▾ All users ▾ All time ▾

-o- Commits on Jan 7, 2025

dondonkim SANGDONKIM committed 19 hours ago	cb7f4c0	🔗	↗
Update README.md ddkim94 authored 20 hours ago	2c4d568	🔗	↗
Initial commit ddkim94 authored 20 hours ago	421b2f7	🔗	↗



# git 대표 오류

**The requested URL returned error: 403**

이 오류는 인증(로그인) 실패 또는 계정이 저장소에 대한 권한이 없음을 나타냅니다. 보통 현재 사용 중인 GitHub 계정이 저장소의 소유자 또는 협력자가 아닌 경우 발생합니다.



## 해결책 1 : 자격 증명 관리자 설정

- 제어판 -> 자격 증명 관리자

The screenshot shows the Windows Credential Manager window titled "자격 증명 관리자". The window is displayed over a blurred background of a web browser. The main content area is titled "제어판 홈" and "사각 증명 관리자". It displays a list of saved credentials under the heading "웹 자격 증명" and "Windows 자격 증명".

Category	Credential Name	Last Modified
Windows 자격 증명	GitHub - https://api.github.com/seoyeoneDA	2024-03-26
	git:https://github.com	2024-05-13
	https://index.docker.io/v1/	2024-06-11
	https://index.docker.io/v1//access-token	2024-06-12
	https://index.docker.io/v1//refresh-token	2024-06-12
	virtualapp/didlogical	2024-06-13
	SSO_POP_Device	오늘

At the bottom of the window, there is a note about managing credentials for cloud services.

You should also become familiar with using RStudio Projects (which are required for version



## 해결책 1 : 자격 증명 관리자 설정

- 일반 자격 증명에 표시된 내용 제거

자격 증명 관리자

제어판 홈 > 제어판 > 사용자 계정 > 자격 증명 관리자

제어판 검색

제어판 홈 웹 자격 증명 Windows 자격 증명

자격 증명 백업(B) 자격 증명 복원(R)

Windows 자격 증명 Windows 자격 증명 추가

Windows 자격 증명이 없습니다.

인증서 기반 자격 증명 인증서 기반 자격 증명 추가

인증서가 없습니다.

일반 자격 증명 일반 자격 증명 추가

GitHub - https://api.github.com/seoyeoneDA

인터넷 또는 네트워크 주소: GitHub - https://api.github.com/seoyeoneDA  
사용자 이름: seoyeoneDA  
암호: \*\*\*\*\*  
지속성: 엔터프라이즈

편집 제거 ←

githttps://github.com 수정한 날짜: 2024-05-13  
https://index.docker.io/v1/ 수정한 날짜: 2024-06-11

참고 항목 사용자 계정



## 해결책 2 : personal token 발급

- github 홈페이지 -> setting

The screenshot shows the GitHub Settings page for the user 'SANGDONKIM'. The main area displays the 'Public profile' settings, including fields for Name (SANGDONKIM), Profile picture (a circular image of a person walking on a path), Bio (Tell us a little bit about yourself), Public email (Select a verified email to display), Pronouns (Don't specify), and URL. On the left, a sidebar menu lists various account management sections: Public profile, Account, Appearance, Accessibility, Notifications, Access (Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, Moderation), and Code, planning, and automation. On the right, a sidebar menu lists links such as Set status, Your profile, Your repositories, Your Copilot, Your projects, Your stars, Your gists, Your organizations, Your enterprises, Your sponsors, Try Enterprise (Free), Feature preview, and Settings (which is highlighted with a red box). Other links include GitHub Website, GitHub Docs, GitHub Support, and GitHub Community.



## 해결책 2 : personal token 발급

- Developer settings -> Personal Access Token -> Tokens(classic)

The screenshot shows the GitHub Developer Settings page. The top navigation bar includes 'Settings' and 'Developer Settings'. The 'Developer Settings' tab is active and highlighted with a red box. On the left, a sidebar lists 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (which is expanded), 'Fine-grained tokens' (with a 'Preview' button), and 'Tokens (classic)' (which is also highlighted with a red box). The main content area is titled 'GitHub Apps' and displays a message: 'No GitHub Apps'. Below this, it says: 'Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API.' It features a green 'New GitHub App' button and a blue 'View documentation' link.



## 해결책 2 : personal token 발급

- Generate new token -> Generate new token(classic)

The screenshot shows the GitHub Developer Settings page under Personal access tokens. A red box highlights the 'Generate new token' button at the top right of the token list. Another red box highlights the 'Generate new token (classic)' option in a dropdown menu, which is described as 'Fine-grained, repo-scoped' and 'For general use'.

Personal access tokens (classic)

Tokens you have generated that can be used to access the [GitHub API](#).

`token` — `admin:org, admin:org_hook, admin:public_key, admin:repo_hook, repo, user, workflow, write:packages`

Expires on *Tue, Mar 25 2025*.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Generate new token ▾

Generate new token (Beta)  
Fine-grained, repo-scoped

Generate new token (classic)  
For general use

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



## 해결책 2 : personal token 발급

- Notes에 적당한 이름 입력 -> Expiration : 90일로 설정 -> 체크 버튼 모두 클릭

**New personal access token (classic)**

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

tokens

What's this token for?

**Expiration \***

90 days The token will expire on Tue, Apr 8 2025

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<hr/>	
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<hr/>	
<input checked="" type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<hr/>	
<input type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry
<hr/>	
<input checked="" type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input checked="" type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input checked="" type="checkbox"/> read:org	Read org and team membership, read org projects



## 해결책 2 : personal token 발급

발급된 token은 다시 확인하려면 재발급 받아야 하므로, 따로 메모장에 저장해야 합니다.

The screenshot shows the GitHub settings interface for managing personal access tokens. The left sidebar includes options for GitHub Apps, OAuth Apps, and Personal access tokens, with the latter being the active section. Under Personal access tokens, there are two sub-options: Fine-grained tokens and Tokens (classic), with Tokens (classic) currently selected. A 'Preview' button is visible next to the fine-grained tokens option. The main content area is titled 'Personal access tokens (classic)' and displays a list of generated tokens. A prominent message at the top of this list reads: 'Tokens you have generated that can be used to access the [GitHub API](#). Make sure to copy your personal access token now. You won't be able to see it again!' Below this message is a single token entry. The token details are as follows:

token	— admin:org, admin:org_hook, admin:public_key, admin:repo_hook, repo, user, workflow, write:packages	Last used within the last week	Delete
Expires on <i>Tue, Mar 25 2025</i> .			

A note at the bottom of the page states: 'Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#)'.



## 해결책 2 : personal token 발급

git push 시 권한 오류가 발생할 경우 personal token을 통해 해결할 수 있습니다.

```
git push https://<your_username>@github.com/<your_username>/<repository>.git
```

```
git push https://SANGDONKIM:token입력@github.com/SANGDONKIM/LS_fork_example.git
```



# 연습 문제 : 나의 저장소 만들기

본 수업 전 공부한 내용을 정리하기 위한 개인 저장소를 만들어보겠습니다.

## 1. 본인 github 계정에 저장소를 만드세요.

- README.md 파일 생성 포함

## 2. 로컬 환경에서 저장소를 복제(clone)하세요.

## 3. 로컬 저장소로 이동한 후, README.md 파일을 열어 아래 내용을 추가하세요:

나의 학습 저장소

- 이 저장소는 Git 및 GitHub 학습 내용을 정리한 개인 학습 저장소입니다.

## 4. 변경한 내용을 저장한 뒤, 아래 명령어로 커밋하세요:



# 연습 문제 : 나의 저장소 만들기

## 5. 원격 저장소에 변경 내용을 푸시하세요.

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'main' (selected), '1 Branch', '0 Tags', a search bar ('Go to file'), an 'Add file' button, a 'Code' dropdown, and an 'About' section. The 'About' section indicates 'No description, website, or topics provided.' It also lists 'Readme', 'Activity', '0 stars', '1 watching', and '0 forks'. Below this, the commit history shows three commits by 'tt' and one by 'SANGDONKIM' (the user). The most recent commit by 'SANGDONKIM' is 'Update README.md' made 'now'. The 'README' file content is displayed below the history:

```
나의 학습 저장소

• 이 저장소는 Git 및 GitHub 학습 내용을 정리한 개인 학습 저장소입니다.
```

해당 이미지를 복사해서 Homework 게시판에 댓글로 업로드해주세요.

게시판 링크 : <https://courses.statisticsplaybook.com/courses/ls/lectures/55316872>