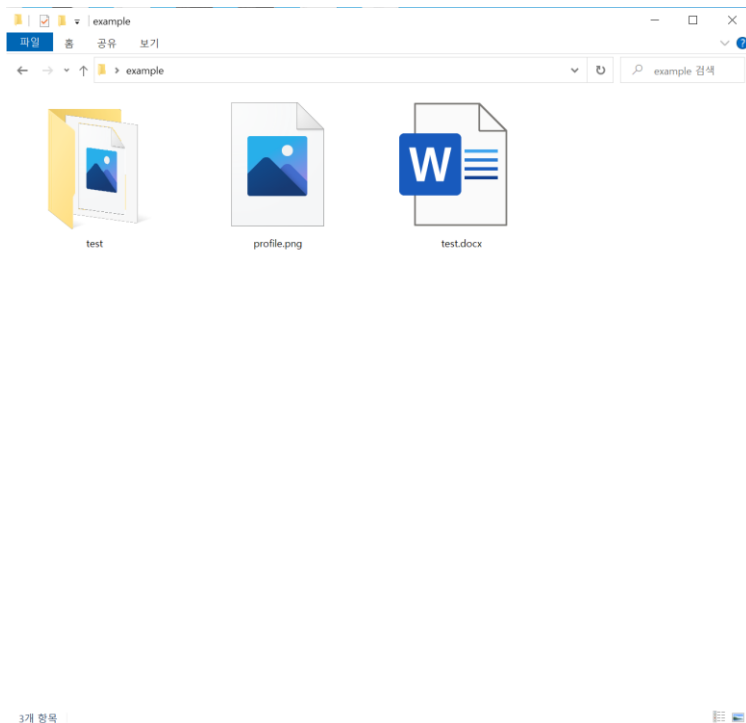


CLI (Command Line Interface)



```
MINGW64/c/Users/lec/Desktop/example
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/example
$ ls
profile.png test/ test.docx
lec@DESKTOP-49D9LQU MINGW64 ~/Desktop/example
$
```



Git

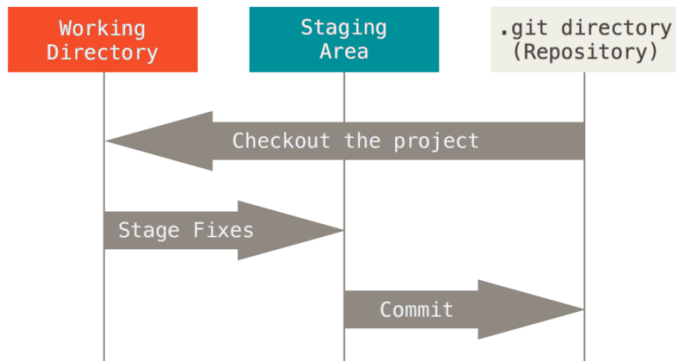
- Git은 분산버전관리시스템으로 코드의 버전을 관리하는 도구
- 2005년 리눅스 커널을 위한 도구로 리누스 토르발스가 개발
- 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율



1. 작업을 하고
2. 변경된 파일을 모아 (add)
3. 버전으로 남긴다. (commit)

기본 흐름

- Git은 파일을 modified, staged, committed로 관리
 - modified : 파일이 수정된 상태 (add 명령어를 통하여 staging area로)
 - staged : 수정한 파일을 곧 커밋할 것이라고 표시한 상태 (commit 명령어로 저장소)
 - committed : 커밋이 된 상태



<https://git-scm.com/book/ko/v2/시작하기-Git-기초>

기본 명령어 – log

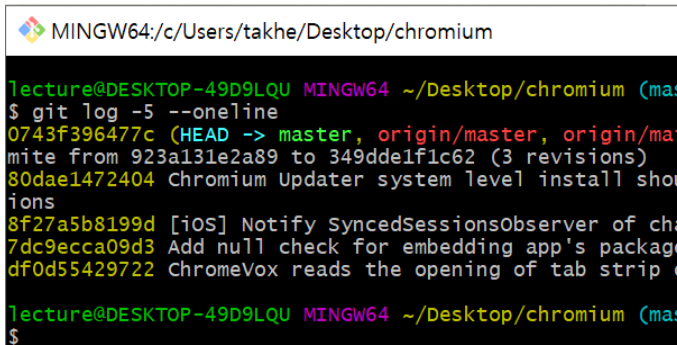
\$ git log

- 현재 저장소에 기록된 커밋을 조회
- 다양한 옵션을 통해 로그를 조회할 수 있음

\$ git log -1

\$ git log --oneline

\$ git log -2 --oneline

A terminal window titled 'MINGW64:/c/Users/takhe/Desktop/chromium' showing the output of the command '\$ git log -5 --oneline'. The output lists five commit hashes and their messages, with the first commit being the HEAD.

```
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/chromium (master)
$ git log -5 --oneline
0743f396477c (HEAD -> master, origin/master, origin/main)
mited from 923a131e2a89 to 349dde1f1c62 (3 revisions)
80dae1472404 Chromium Updater system level install show
ions
8f27a5b8199d [iOS] Notify SyncedSessionsObserver of cha
7dc9ecca09d3 Add null check for embedding app's package
df0d55429722 ChromeVox reads the opening of tab strip c
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/chromium (master)
$
```

기본 명령어 – status

\$ git status

- Git 저장소에 있는 파일의 상태를 확인하기 위하여 활용
 - 파일의 상태를 알 수 있음
 - Untracked files
 - Changes not staged for commit
 - Changes to be committed
 - Noting to commit, working tree clean

```
MINGW64/c/Users/takhe/Desktop/git-1
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git add b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   b.txt

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git commit -m 'Add b.txt'
[master 9f47127] Add b.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 b.txt

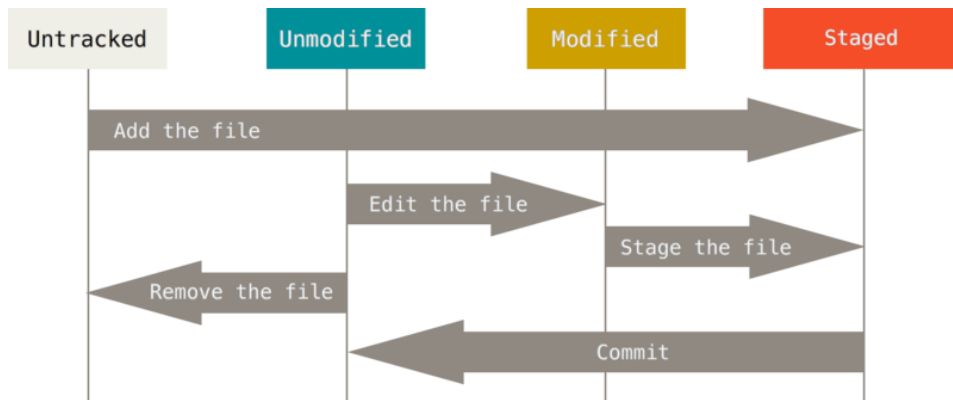
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git status
On branch master
nothing to commit, working tree clean

lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ |
```

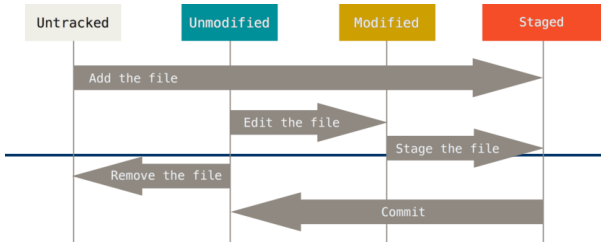
파일 라이프사이클

- Status로 확인할 수 있는 파일의 상태
 - Tracked : 이전부터 버전으로 관리되고 있는 파일
 - Unmodified : git status에 나타나지 않음
 - Modified : Changes not staged for commit
 - Staged : Changes to be committed
 - Untracked : 버전으로 관리된 적 없는 파일 (파일을 새로 만든 경우)

파일 라이프사이클



<https://git-scm.com/book/ko/v2/Git의-기초-수정하고-저장소에-저장하기>



Git 기초 명령어

명령어	내용
git init	로컬 저장소 생성
git add <파일명>	특정 파일/폴더의 변경사항 추가
git commit -m '<커밋메시지>'	커밋 (버전 기록)
git status	상태 확인
git log	버전 확인

Git 설정 파일 (config)

- 사용자 정보 (commit author) : 커밋을 하기 위해 반드시 필요
 - `git config --global user.name "username"`
 - Github에서 설정한 username으로 설정
 - `git config --global user.email "my@email.com"`
 - Github에서 설정한 email로 설정
- 설정 확인
 - `git config -l`
 - `git config --global -l`
 - `git config user.name`

git 실습 1. 저장소 만들고 첫번째 버전 기록하기

- 1) 바탕화면에 test 폴더를 만들고 git 저장소를 만들기
- 2) a.txt 파일 넣고 커밋하기
- 3) 임의의 파일을 만들고 커밋하기
- 4) a.txt 파일 수정하고 커밋하기

각 단계별로 status와 log를 보세요.

- 주의
 - 어떠한 파일도 생성하지 않으면 당연히도 버전을 만들 대상이 없습니다.

(추가) 디렉토리에 대한 이해

- CLI에서 현재 폴더의 목록을 보면 다음과 같다.
 - . : 현재 디렉토리
 - 그래서 git add . 이 현재 폴더에 대한 모든 파일의 변경사항을 add 하는 것!
 - .. : 상위 디렉토리

```
$ ls -al
total 16
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 ./ # <---
drwxr-xr-x 1 lec 197121 0  7월 15 11:32 ../ # <---
drwxr-xr-x 1 lec 197121 0  7월 15 11:26 .git/
-rw-r--r-- 1 lec 197121 0  7월 15 11:26 b.txt
-rw-r--r-- 1 lec 197121 0  7월 15 11:26 c.txt
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 images/
drwxr-xr-x 1 lec 197121 0  7월 15 11:54 my_folder/
```

(추가) 디렉토리에 대한 이해 (상대 경로 / 절대 경로)

- README.md에서 b.png를 활용하기 위해서는?
 - 절대 경로 : C:/TIL/images/markdown/b.png
 - 상대 경로 : ./images/markdown/b.png

```
C: /  
  TIL/  
    images/  
      markdown/  
        a.png  
        b.png  
      README.md
```

(추가) 디렉토리에 대한 이해 (상대 경로 / 절대 경로)

- style.css에서 a.png를 활용하기 위해서는?
 - style.css 위치는 my_web의 css!
 - 즉, 상대 경로로 표현하면 ../images/a.png

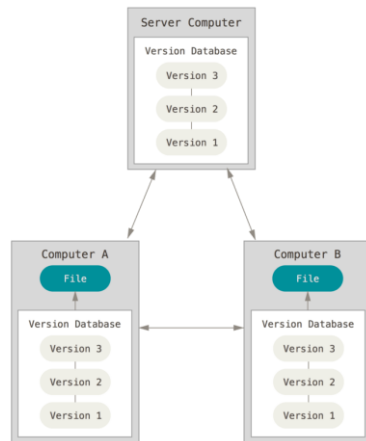
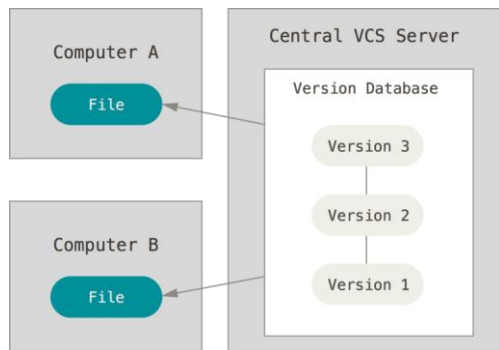
```
my_web/  
  images/  
    a.png  
  css/  
    style.css  
  index.html
```


원격저장소 활용하기

GITHUB

분산버전관리시스템(DVCS)

- 중앙집중식버전관리시스템은 중앙에서 버전을 관리하고 파일을 받아서 사용
- 분산버전관리시스템은 원격 저장소(remote repository)를 통하여 협업하고, 모든 히스토리를 클라이언트들이 공유

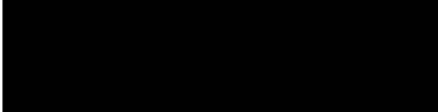


원격저장소 (Remote Repository)

- 네트워크를 활용한 저장소

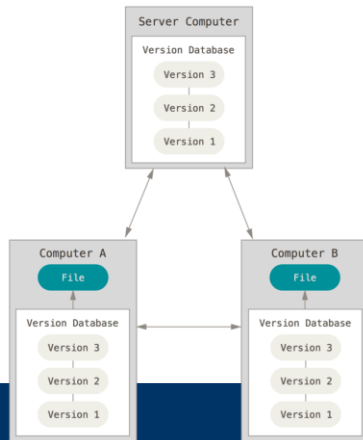




Git은 을 관리한다.

Git은 버전(커밋)을 관리한다.

GitHub도 버전(커밋)을 관리한다.



원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전을 원격저장소로 보낸다.



원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전(커밋)을 원격저장소로 보낸다.

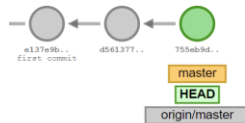
Local Repository
HEAD: master



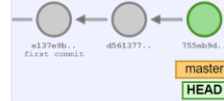
origin



Local Repository
HEAD: master



origin



원격저장소 (Remote Repository) 기본 흐름

- 로컬 저장소의 버전(커밋)을 원격저장소로 보낸다.

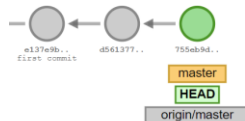
Local Repository
HEAD: master



origin

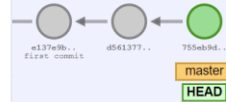


Local Repository
HEAD: master



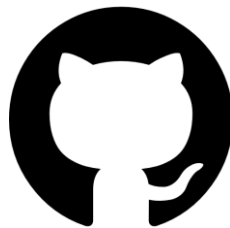
\$ git push

origin



원격저장소 (Remote Repository) 기본 흐름

- 원격저장소의 버전(커밋)을 로컬 저장소로 가져온다.



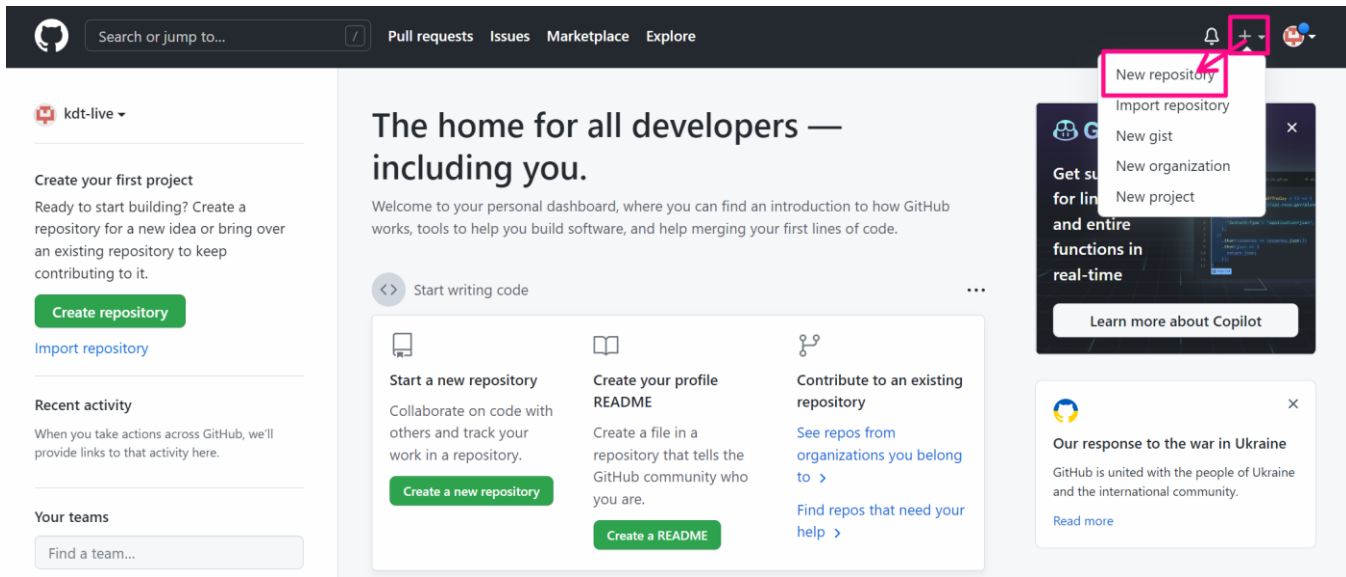
\$ git pull

실제로 활용해보자!

원격저장소를 만들고,
로컬저장소의 커밋을 push한다.
(로컬저장소에 원격 저장소 정보는 필수!)

1. GitHub에서 원격 저장소 만들기 (1/3)

1. New Repository



1. GitHub에서 원격 저장소 만들기 (2/3)

2. 저장소 설정하기

The screenshot shows the GitHub 'Create a new repository' page. The page has a dark header with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is titled 'Create a new repository' and includes a sub-header 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.'.

The form contains several sections:

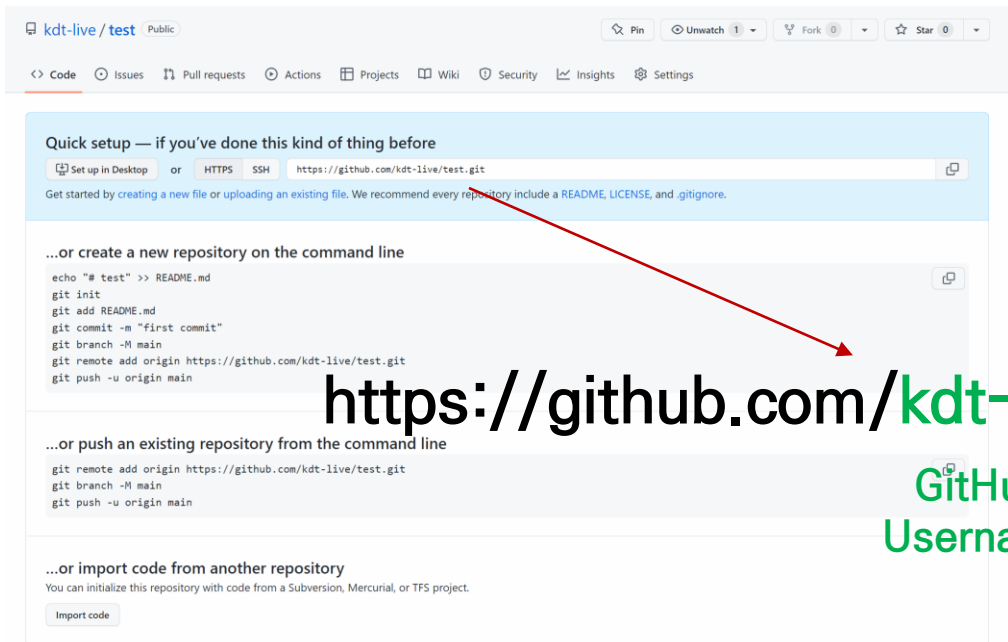
- Owner:** A dropdown menu showing 'kdt-live'.
- Repository name:** A text input field containing 'test'. This field is highlighted with a pink box and labeled '1. repository 이름 설정'.
- Description (optional):** A text input field. This field is highlighted with a pink box and labeled '2. 저장소 설명(옵션)'.
- Visibility:** Two radio buttons: 'Public' (selected) and 'Private'. The 'Public' option is highlighted with a pink box and labeled '3. 공개 여부 설정'.
- Initialize this repository with:** A section with a checkbox for 'Add a README file'.
- Add .gitignore:** A section with a dropdown menu for 'gitignore template' set to 'None'.
- Choose a license:** A section with a dropdown menu for 'License' set to 'None'.
- Create repository:** A green button at the bottom. This button is highlighted with a pink box and labeled '4. 저장소 생성'.

Annotations and red text on the right side of the page:

- A red arrow points from the 'Add .gitignore' section to the text: 'Git/GitHub을 이해하기 전에는 설정을 하지 않습니다. 체크 없이 모두 None!'

1. GitHub에서 원격 저장소 만들기 (3/3)

3. 확인하기



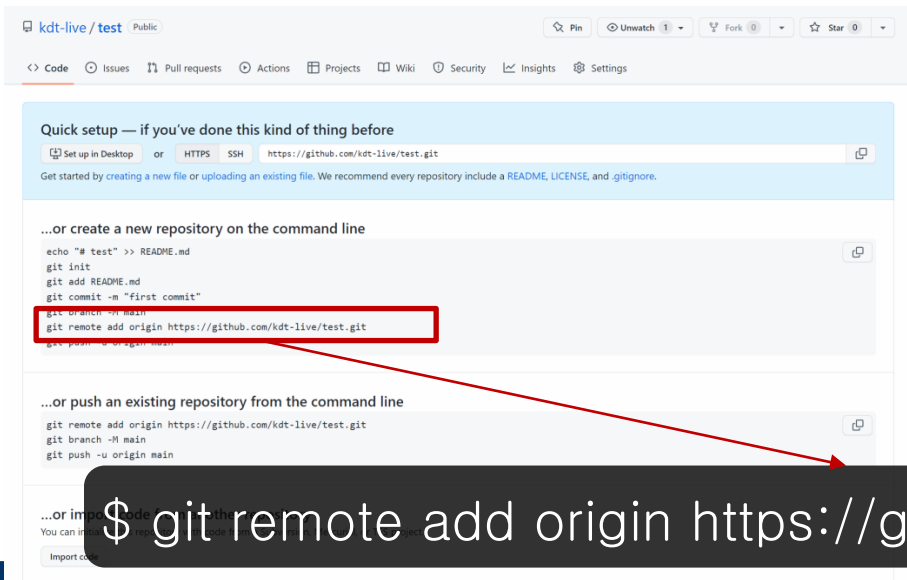
`https://github.com/kdt-live/test.git`

GitHub
Username

저장소
이름

2. 원격저장소 경로 설정

- 원격 저장소 정보를 로컬 저장소에 추가
- 로컬 저장소에는 한번만 설정 해주면 된다.**



Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/kdt-live/test.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kdt-live/test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/kdt-live/test.git
git branch -M main
git push -u origin main
```

...or import code from a different version control system

You can import code from a Subversion repository into a new Git repository or you can use `git svn` to manage existing Subversion code.

Import code from a Subversion repository

ProTip! Use the URL for this page when adding GitHub as a remote.

\$ git remote add origin https://github.com/kdt-live/test.git

2. 원격저장소 경로 설정

- 원격 저장소 정보를 로컬 저장소에 추가
- **로컬 저장소에는 한번만 설정 해주면 된다.**

```
$ git remote add origin https://github.com/kdt-live/test.git
```

원격저장소 추가해 Origin
 으로

GitHub
Username

저장소
이름

2. 원격저장소 정보 확인

\$ git remote -v

- 원격 저장소의 정보를 확인함



```
MINGW64/c/Users/hphk/Desktop/test
hphk@DESKTOP-LGET50N MINGW64 ~/Desktop/test (master)
$ git remote -v
origin https://github.com/kdt-live/test.git (fetch)
origin https://github.com/kdt-live/test.git (push)
hphk@DESKTOP-LGET50N MINGW64 ~/Desktop/test (master)
$
```

원격저장소 이름 (origin)

URL

3. 원격저장소 활용 명령어 - push

\$ git push <원격저장소이름> <브랜치이름>

- 원격 저장소로 로컬 저장소 변경 사항(커밋)을 올림(push)
- 로컬 폴더의 파일/폴더가 아닌 저장소의 버전(커밋)이 올라감

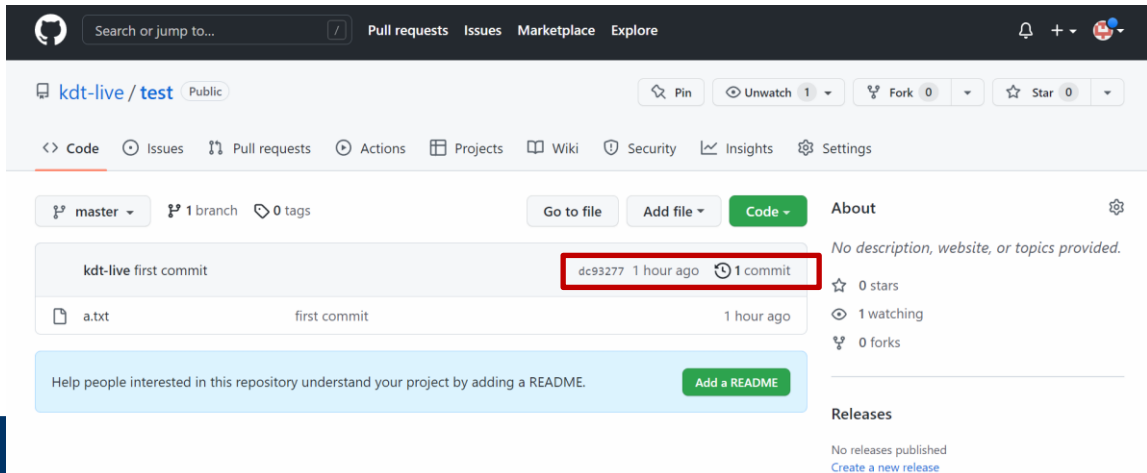
A terminal window titled 'MINGW64:/c/Users/takhe/Desktop/git-1' showing the execution of the 'git push' command. The output indicates a successful push to the 'origin master' branch, including details about object enumeration, compression, and writing. The prompt returns to the shell after the push is complete.

```
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 623 bytes | 311.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/edutak/test-remote.git
 * [new branch]      master -> master
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$
```

3. 원격저장소 활용 명령어 - push

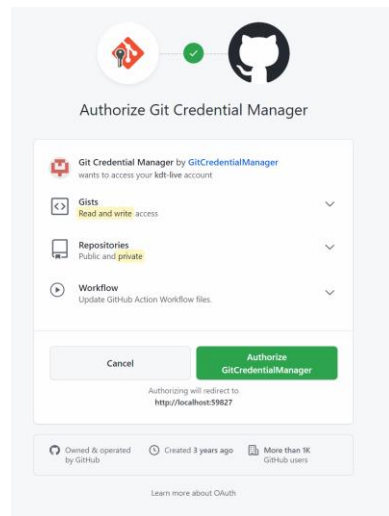
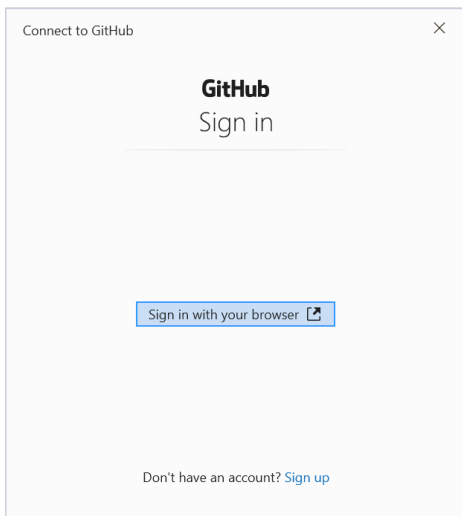
\$ git push <원격저장소이름> <브랜치이름>

- 원격 저장소로 로컬 저장소 변경 사항(커밋)을 올림(push)
- 로컬 폴더의 파일/폴더가 아닌 저장소의 버전(커밋)이 올라감



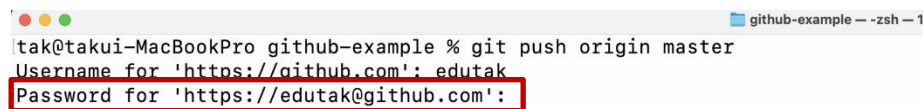
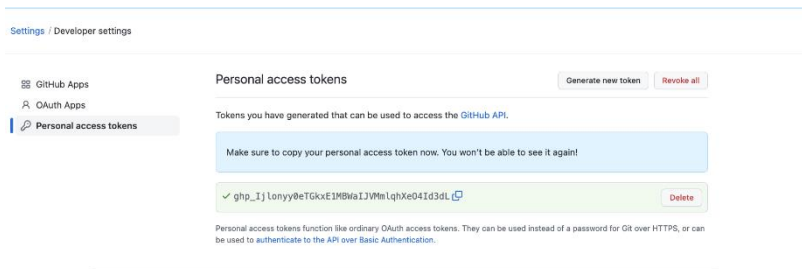
3. push 주의 사항

- Push할 때는 인증 정보가 필수적입니다.
- 윈도우는 아래의 화면에서 인증을 하여야 합니다. (크롬 브라우저 활용)



3. push 주의 사항

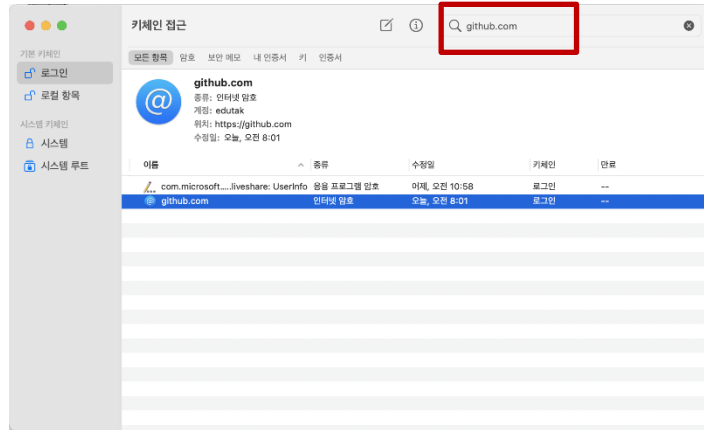
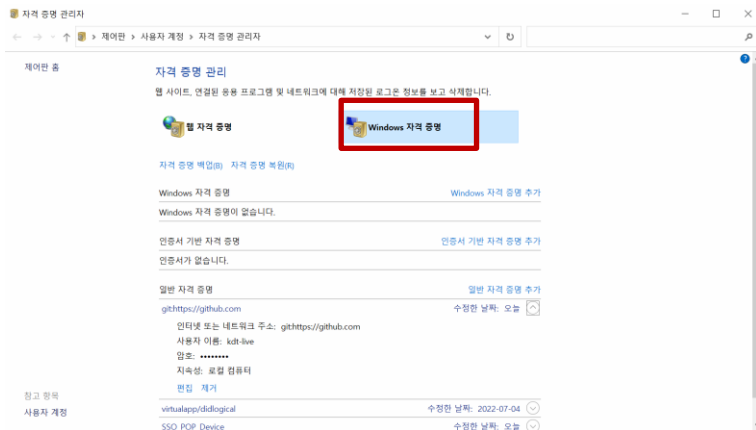
- Push할 때는 인증 정보가 필수적입니다.
- 맥은 토큰을 발급 받아 비밀번호로 활용합니다.



GitHub 로그인 비밀번호가 아닌 Token값!

3. push 주의 사항

- Push가 Authentication failed되는 경우 인증 정보를 확인 부탁드립니다.
 - 윈도우 : 자격증명관리자
 - 맥 : 키체인 접근



원격저장소
업데이트된 커밋을
가져오고 싶어요

원격저장소 활용 명령어 - pull

\$ git pull <원격저장소이름> <브랜치이름>

- 원격 저장소로부터 변경된 내역을 받아와서 이력을 병합함

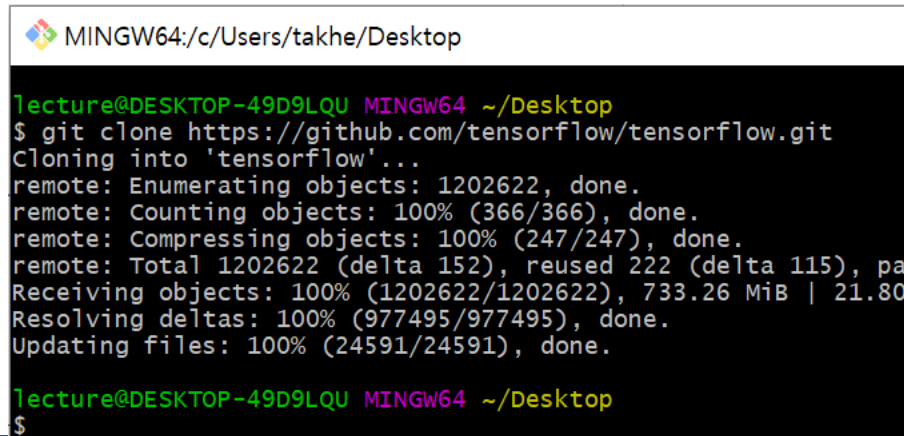
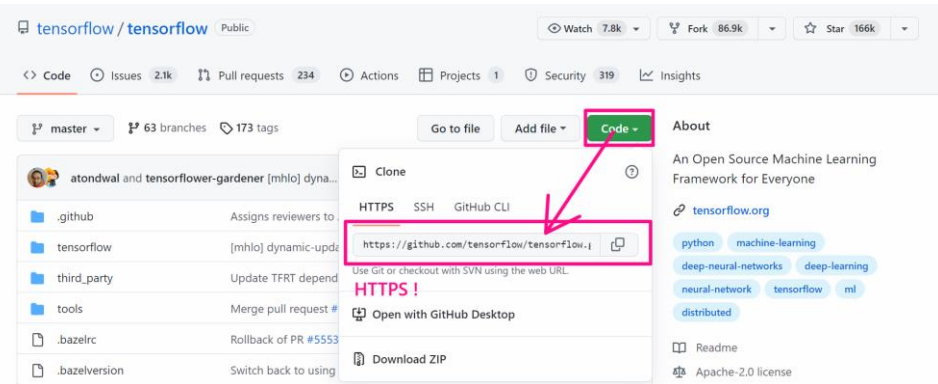
```
MINGW64:/c/Users/takhe/Desktop/git-1
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 718 bytes | 55.00 KiB/s, done.
From https://github.com/edutak/test-remote
* branch      master      -> FETCH_HEAD
   8499447..8196332 master  -> origin/master
Updating 8499447..8196332
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
(base)
lecture@DESKTOP-49D9LQU MINGW64 ~/Desktop/git-1 (master)
$ |
```

**원격저장소 프로젝트를
시작하고 싶어요**

원격저장소 활용 명령어 - clone

\$ git clone <원격저장소주소>

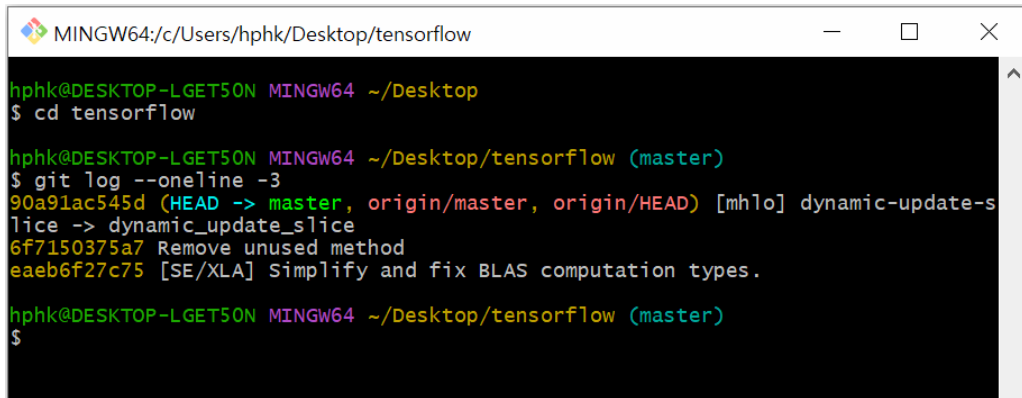
- 원격 저장소를 복제하여 모든 버전을 가져옴
- 원격 저장소 이름의 폴더로 이동해서 활용함



원격저장소 활용 명령어 - clone

\$ git clone <원격저장소주소>

- 원격 저장소를 복제하여 모든 버전을 가져옴
- 원격 저장소 이름의 폴더로 이동해서 활용함



```
MINGW64:/c/Users/hphk/Desktop/tensorflow
hphk@DESKTOP-LGET50N MINGW64 ~/Desktop
$ cd tensorflow

hphk@DESKTOP-LGET50N MINGW64 ~/Desktop/tensorflow (master)
$ git log --oneline -3
90a91ac545d (HEAD -> master, origin/master, origin/HEAD) [mhlo] dynamic-update-slice -> dynamic_update_slice
6f7150375a7 Remove unused method
eaeb6f27c75 [SE/XLA] Simplify and fix BLAS computation types.

hphk@DESKTOP-LGET50N MINGW64 ~/Desktop/tensorflow (master)
$
```

Clone과 Pull의 차이점은?

Clone : 원격저장소 복제

Pull : 원격저장소 커밋 가져오기

로컬에서 새로운 프로젝트의 시작 ?

로컬에서 새로운 프로젝트의 시작 ?

`git init`

원격에 있는 프로젝트 시작?

원격에 있는 프로젝트 시작?

git clone

협업 프로젝트, 외부 오픈소스 참여
Git 저장소를 GitHub에서 생성 후 시작 등..

프로젝트 개발 중 다른 사람 커밋 받아오기?

프로젝트 개발 중 다른 사람 커밋 받아오기?

`git pull`

내가 한 로컬 프로젝트 개발 공유?

내가 한 로컬 프로젝트 개발 공유?

git push

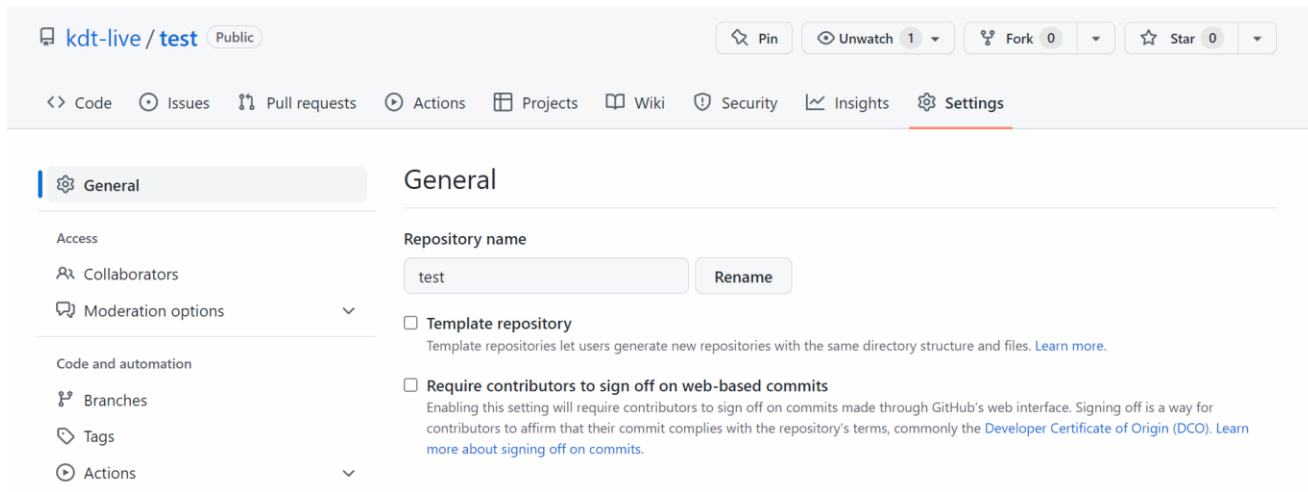
원격저장소 설정 기본 명령어

명령어	내용
git clone <url>	원격 저장소 복제
git remote -v	원격저장소 정보 확인
git remote add <원격저장소> <url>	원격저장소 추가 (일반적으로 origin)
git remote rm <원격저장소>	원격저장소 삭제
git push <원격저장소> <브랜치>	원격저장소에 push
git pull <원격저장소> <브랜치>	원격저장소로부터 pull

기타 주요 저장소 관리

저장소 이름 변경

- Settings > General > Repository name
- 저장소 이름 변경시 원격저장소 URL이 변경되어 로컬 설정 변경이 필수적입니다.



저장소 Public/Private 전환 및 삭제

- Settings > General > 하단부 Danger Zone

Danger Zone

Change repository visibility

This repository is currently public.

[Change visibility](#)

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

[Transfer](#)

Archive this repository

Mark this repository as archived and read-only.

[Archive this repository](#)

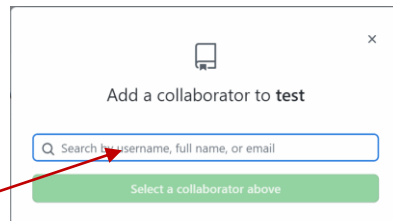
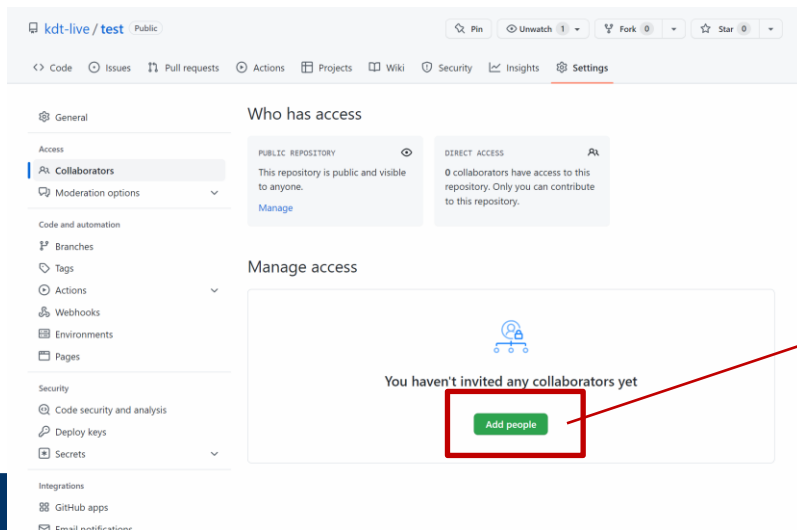
Delete this repository

Once you delete a repository, there is no going back. Please be certain.

[Delete this repository](#)

저장소 접근 관리

- Settings > Collaborators
- 저장소에 push 권한은 collaborator에만 있습니다.
 - 초대를 받은 사람은 이메일로 초대가 되며, 승낙한 경우 공동 작업이 가능합니다.



Push 실패

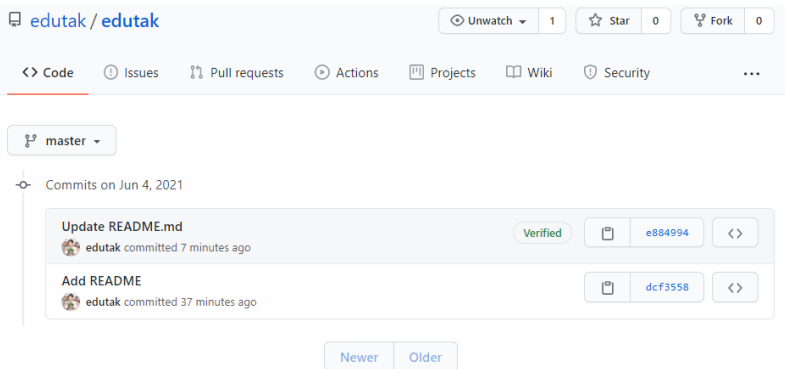
Push 실패

- 협업을 하다보면 아래의 메시지를 확인하게 된다.

```
student@M503INS MINGW64 ~/Desktop/edutak (master)
$ git push origin master
To https://github.com/edutak/edutak.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/edutak/edutak.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.
```

Push 실패

- 로컬과 원격 저장소의 커밋 이력이 다른 경우 발생한 것이다.











```
student@M503INS MINGW64 ~/Desktop/edutak (master)
$ git log --oneline
8946b62 (HEAD -> master) Update 경 력
dcf3558 (origin/master) Add README
```

Push 실패

- 1. 원격저장소의 커밋을 원격저장소로 가져와서(pull)
- 2. 로컬에서 두 커밋을 병합 (추가 커밋 발생)
 - 동시에 같은 파일이 수정된 경우 merge conflict가 발생하나 이 부분은 브랜치 학습
- 3. 다시 GitHub으로 push

Commits on Jun 4, 2021

Merge branch 'master' of https://github.com/edutak/edutak edutak committed 2 minutes ago	 b7b72e1	
Update 경력 edutak committed 10 minutes ago	 3946b62	
Update README.md edutak committed 12 minutes ago	Verified  e884994	
Add README edutak committed 42 minutes ago	 dcf3558	

Git 파일 관리 심화

GITIGNORE

버전관리랑 상관 없는 파일?



secret.csv

어떻게 관리해야할까?

.gitignore

- 일반적인 개발 프로젝트에서 버전 관리를 별도로 하지 않는 파일/디렉토리가 발생한다.
- Git 저장소에 .gitignore 파일을 생성하고 해당 내용을 관리한다.
- 작성 예시
 - 특정 파일 : a.txt (모든 a.txt), test/a.txt (테스트 폴더의 a.txt)
 - 특정 디렉토리 : /my_secret
 - 특정 확장자 : *.exe
 - 예외 처리 : !b.exe
- **주의! 이미 커밋된 파일은 반드시 삭제를 하여야 .gitignore로 적용됩니다.**
 - 따라서, 프로젝트 시작전에 미리 설정하시기 바랍니다 ☺

**일반적으로
어떤 파일들이 있을까?**

.gitignore

- 개발 언어 (<https://github.com/github/gitignore>)
 - 예시) 파이썬 : venv/ , 자바스크립트 : node_modules/
- 개발 환경
 - 운영체제 (windows, mac, linux)
 - 텍스트 에디터 / IDE (visual studio code 등)

