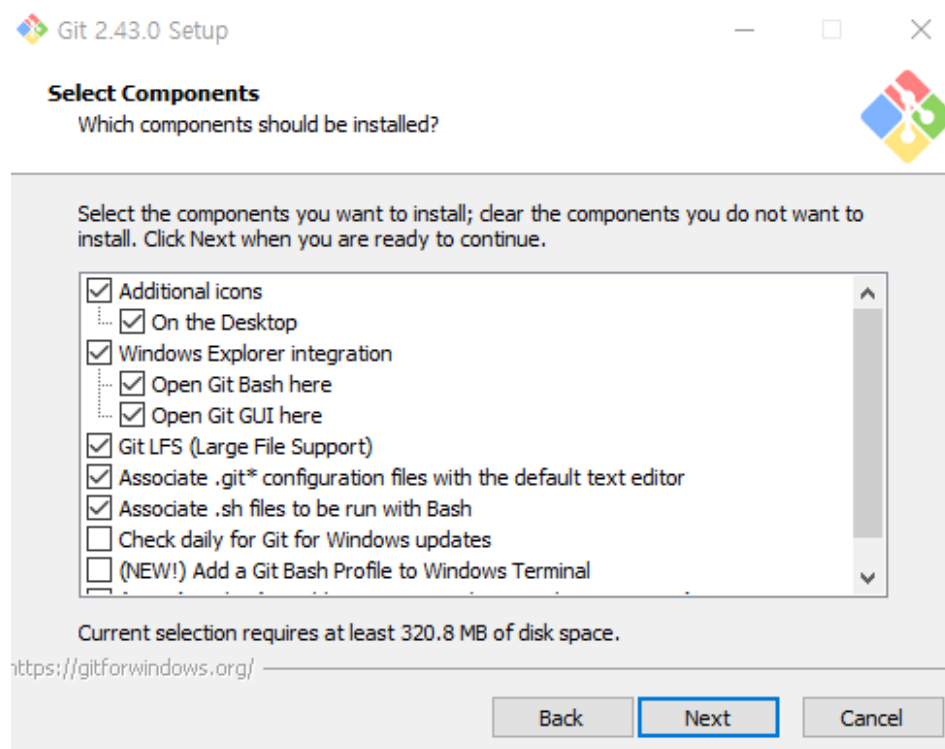
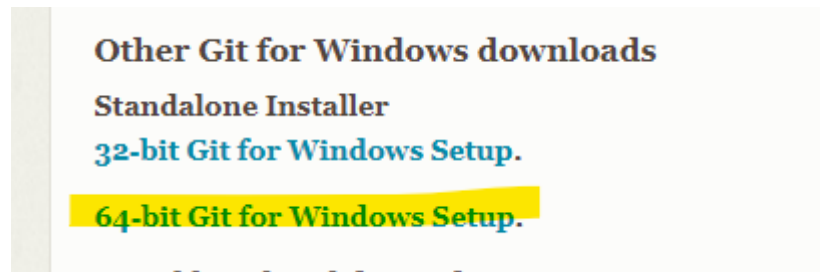


1월 16일 화요일
AM

<https://git-scm.com/download/win>



CSS - Flex, Grid

<https://github.com/> > 로그인 > Top Repositories New

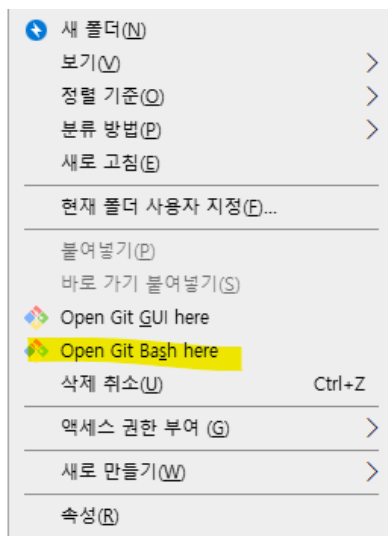
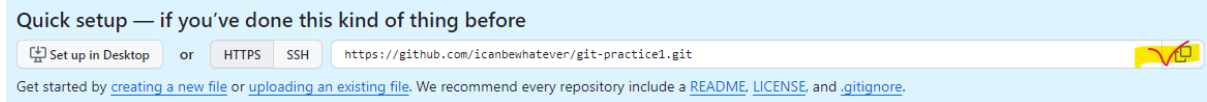
Git

```
human-20@DESKTOP-85ADT1V MINGW64 ~  
$ git
```

내 폴더 위치 검색 pwd 또는 파일탐색기 창 띄우기: 윈도우+e

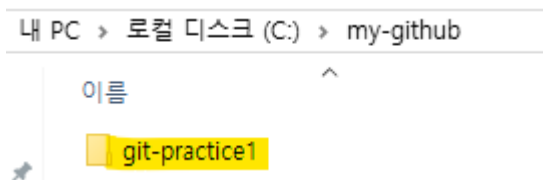
```
human-20@DESKTOP-85ADT1V MINGW64 ~  
$ pwd  
/c/Users/human-20
```

로컬디스크와 연결하기 -



로컬디스크에 폴더만들고 > 오른쪽 마우스 클릭시 나온다


```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github  
$ git clone https://github.com/icanbewhatever/git-practice1.git  
Cloning into 'git-practice1'...  
warning: You appear to have cloned an empty repository.
```



폴더가 자동적으로 생성된다

명령어 배우기

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

 Set up in Desktop or **HTTPS** SSH <https://github.com/icanbewhat>

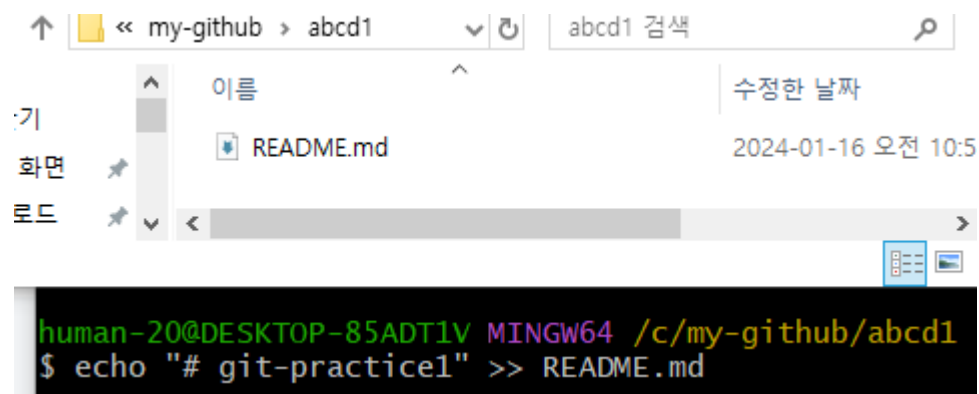
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend even


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

```
echo "# git-practice1" >> README.md
git init
git add README.md
```

로컬디스크 > abcd1

리눅스 명령어 > **echo "# git-practice1" >> README.md**



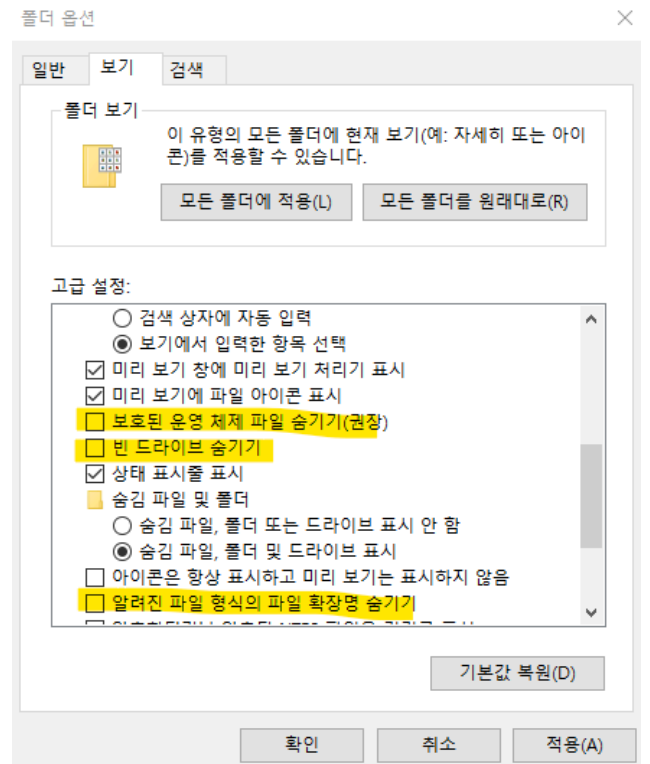
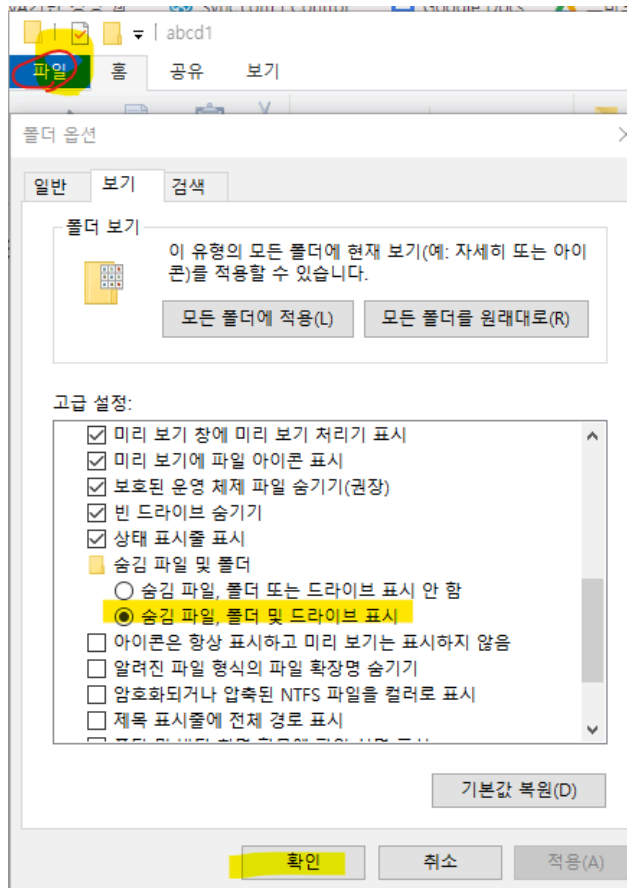
↑  << my-github > abcd1 | abcd1 검색

이름	수정한 날짜
README.md	2024-01-16 오전 10:5

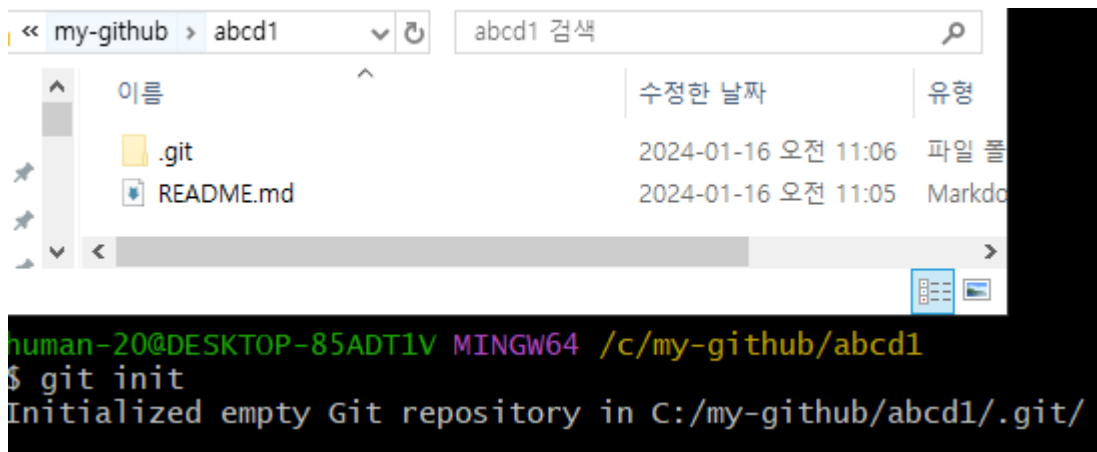
```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abcd1
$ echo "# git-practice1" >> README.md
```

또는 아래처럼 만들수 있다 > **echo "하하호호" >> README.md**

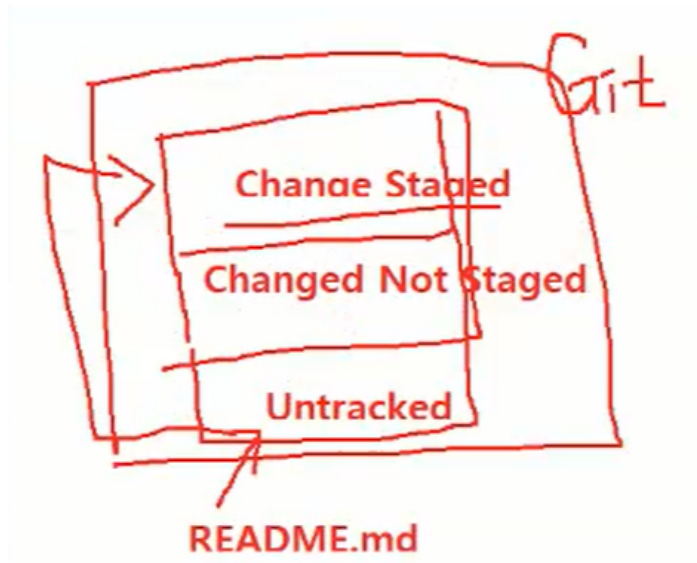
```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abcd1
$ echo "하 하 호 호 " >> README.md
```



git 폴더 만들어주기 > \$ git init



.git (파일관리)과 README.md 두 파일이 있어야지만 실행이 가능하다.



README를 만들면 먼저 **Untracked** 상태가 된다 > 준비만(대기) 한 상태
git commit -m "first commit" > 메시지 작성 코드

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

```
echo "# git-practice1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/icanbewhatever/git-practice1.git
git push -u origin main
```

```
MINGW64:/c/my-github/abc
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

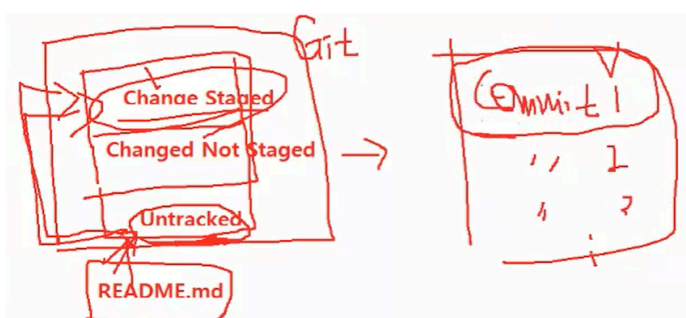
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$
```

git status
git add README.md
git status

커밋을 할 경우

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

```
echo "# git-practice1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/icanbewhatever/git-practice1.git
git push -u origin main
```



git commit -m "first commit"

커밋 에러> 식별자가 필요하다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git commit -m "첫 번째 커밋입니다."
Author identity unknown

*** please tell me who you are.

Run

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'human-20@DESKTOP-85ADT1V.(none)')
```

임의의 이메일과 이름을 넣어준다

```
*** Please tell me who you are.

Run

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'human-20@DESKTOP-85ADT1V.(none)')

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git config --global user.email "student@human.com"

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git config --global user.name "luna"
```

그러면 커밋할 수 있는 상태가 만들어진다

다시 커밋 > **git commit -m "첫번째 커밋이다"**

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/abc (master)
$ git commit -m "첫 번째 커밋이다"
[master (root-commit) 495df63] 첫 번째 커밋이다
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

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

```
echo "# git-practice1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/icanbewhatever/git-practice1.git
git push -u origin main
```

git remote add... 복붙 입력

git push.... 복붙 입력

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git remote add origin https://github.com/icanbewhatever/git-practice1.git
error: remote origin already exists.

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 214 bytes | 214.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/icanbewhatever/git-practice1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$
```

<https://github.com/> 로그인 창이 뜨고 authorize 해주면 된다.

다시복습 > github에 **Repositories** 폴더 생성후 로컬 **C** 에 파일생성해 연결하기

1. \$ git clone https://github.com/icanbewhatever/git-practice1.git
2. \$ git init – git라는 파일을 넣어준다
3. echo "# git-practice1" >> README.md
4. git status
5. git add. (.이 모든 파일을 선택해준다) / git add 파일명
6. git status
7. git commit -m "폴더생성테스트"
(쓰는이유> 커밋 메시지를 강제로 쓰게끔 git이 만들어지기 때문에 git에서 commit은 **git log**라는 명령어가 있듯이 commit으로 파일의 변경시점을 알 수있어서기도 하다.)
8. git push -u origin main 또는 git push


```

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ echo "헬로우" >> README2.md

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git add README2.md
warning: in the working copy of 'README2.md', LF will be replaced by
CRLF the next time Git touches it

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   README2.md

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git commit -m "두 번째"
[main 6f449a3] 두 번째
 1 file changed, 1 insertion(+)
 create mode 100644 README2.md

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/icanbewhatever/git-practice1.git
   1acf640..6f449a3  main -> main
branch 'main' set up to track 'origin/main'.

```

파일 삭제하기?

git add 파일명 > git commit -m "아무거나" > git push

파일 두개 이상 삭제시 git add .

만들어진 readme.md파일은 기본 `chaged not staged`에 있다 이것을 `change staged`에 넣어 커밋 후 github로 넘겨줘야 한다

1. 먼저 폴더에서 원하는 파일을 삭제, 그렇지만 github에서는 삭제처리가 되지 않았다
2. git status 를 입력해 deleted: README2.md 확인
3. git add README2.md
4. git commit -m "README2" (deleted file 이라고 써줄 필요 없이 그냥 1번 실행)
5. git push

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:      README2.md

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git add README2.md
fatal: pathspec 'README2.md' did not match any files

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git commit -m "README2 deleted file"
[main 384e429] README2 deleted file
 1 file changed, 1 deletion(-)
 delete mode 100644 README2.md

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 234 bytes | 234.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/icanbewhatever/git-practice1.git
 6f449a3..384e429  main -> main
```

PM

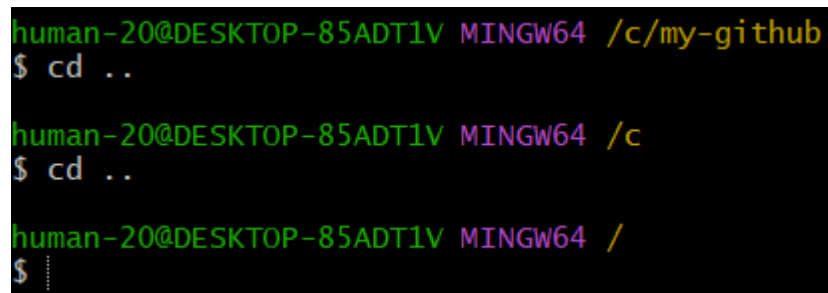
ls 명령어: 파일리스트를 알려준다

ls -al : 숨긴 파일까지 다 알려준다



cd 명령어: 폴더 디렉토리를 알려준다

\$ cd git-practice1 : git-practice1 폴더로 이동



메모장 만들기

<https://notepad-plus-plus.org/downloads/>

Download Notepad++ v8.6.2

git status - 빈 폴더는 인식을 못 한다

git add . - 점이 모든 파일을 선택해준다 현재 디렉토리(폴더)의 모든 새로 생성된 파일을 commit할 수 있도록 Staged Area에 등록하는 명령어

git log - 선택한 폴더의 활동내역 **q** - 나가기

다른사람계정으로 하기

<https://github.com/pktjesus/git-practic2>

git pull 폴더의 새로고침 버튼 후 -> **:q** 엔터 하면 모든 폴더가 리로드 된다

<https://tortoisegit.org/>

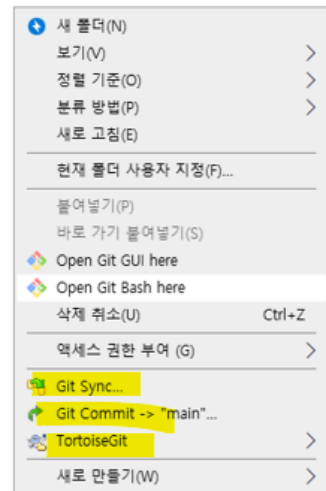
for 64-bit Windows

Download TortoiseGit 2.15.0 64-bit
(21.5 MiB)

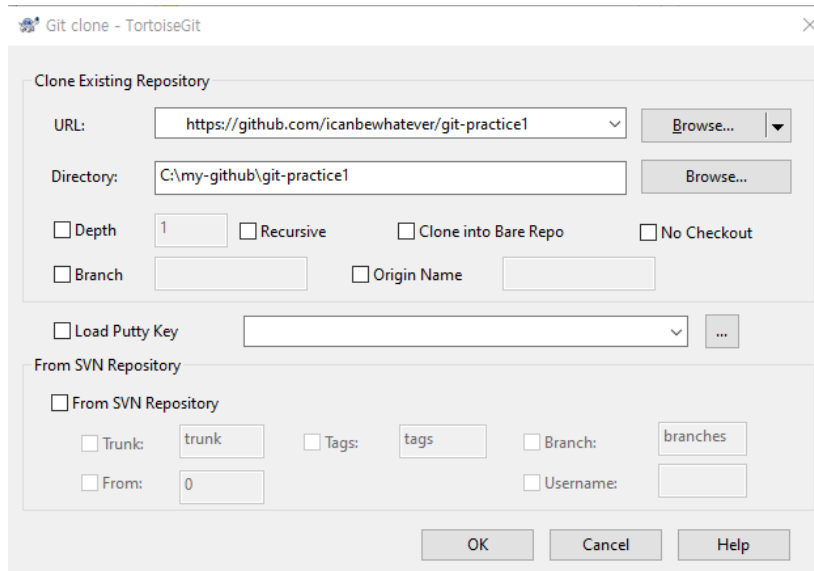
다운로드

xt
1E.md
1E2.md

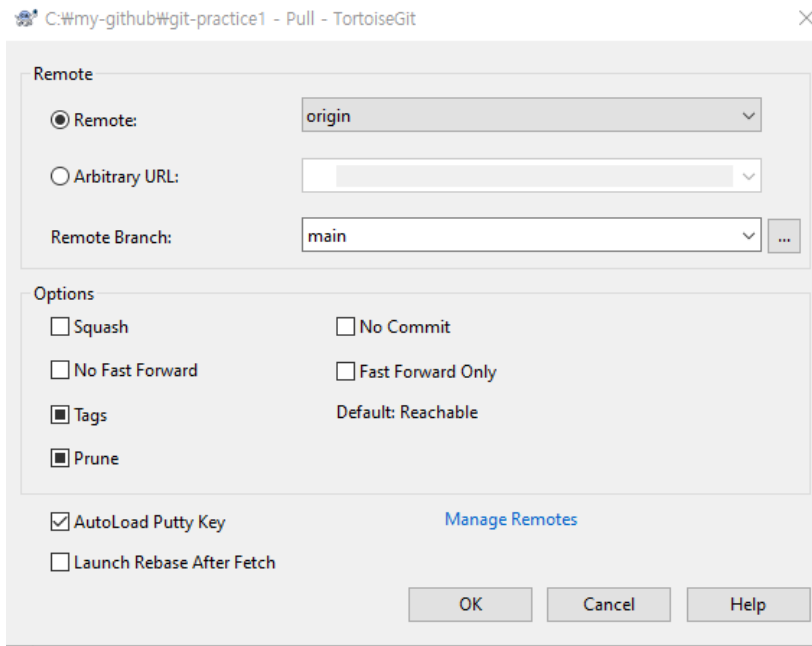
2024-01-16 오후 4:13 엑스드 문서
2024-01-16 오후 2:28 Markdown 원본
2024-01-16 오후 2:28 Markdown 원본



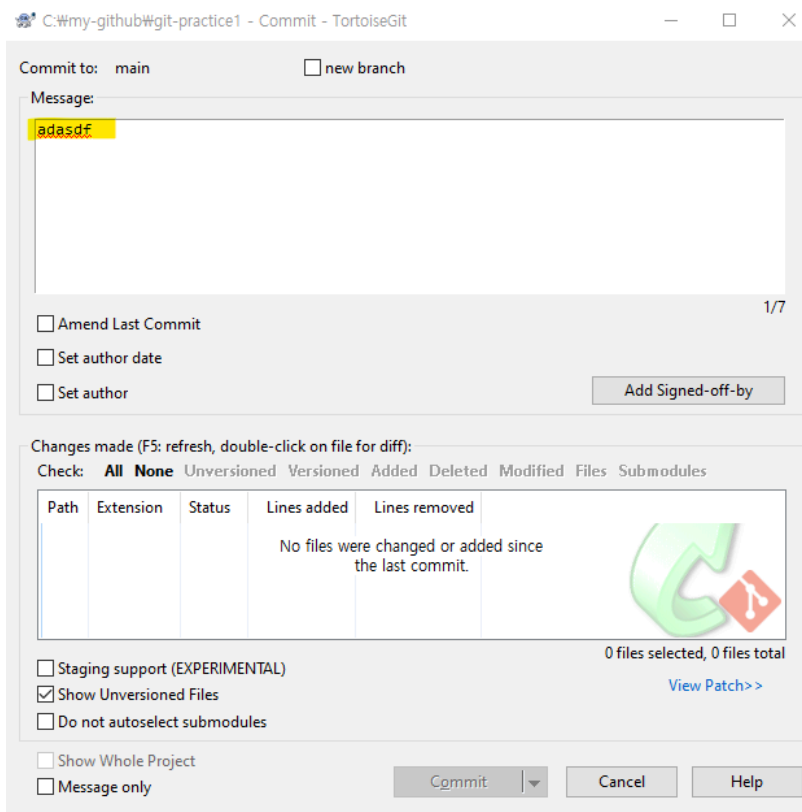
.git 가 있는 폴더에서 오른쪽 마우스 클릭시 >



이것과 **git clone** 과 같다



git pull과 같다



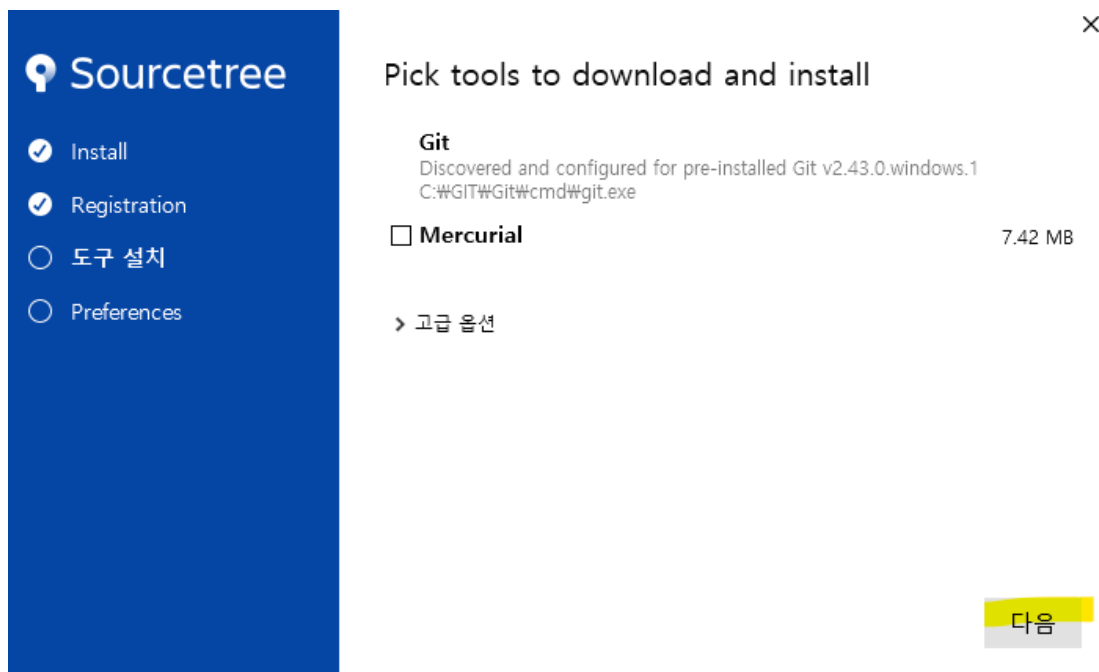
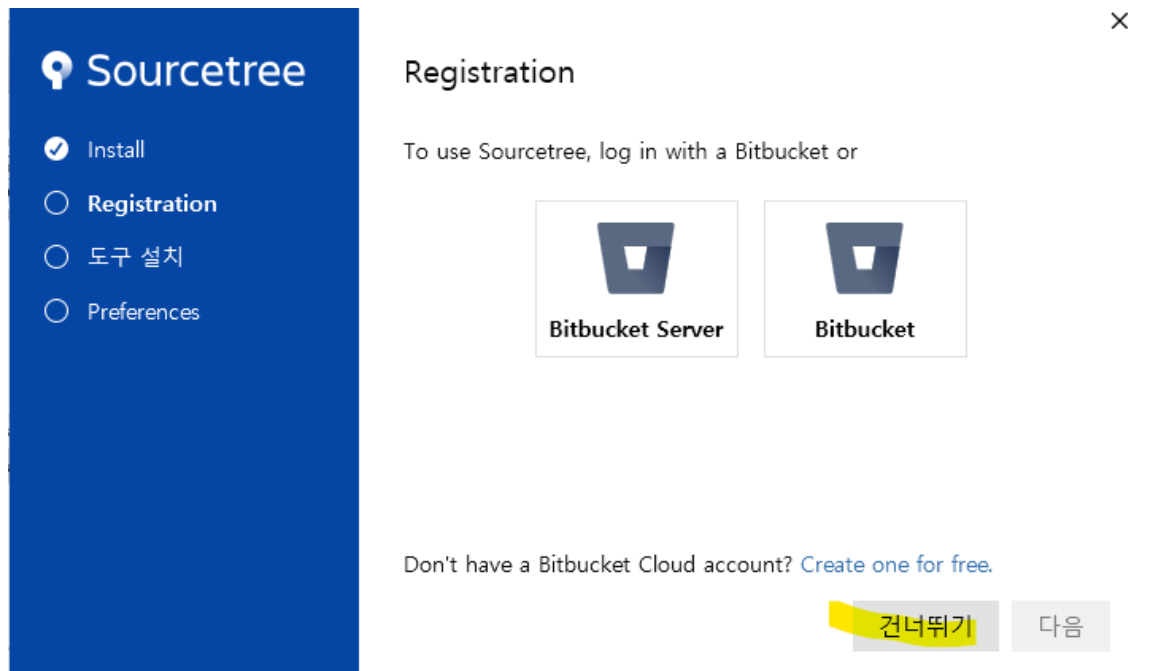
git commit -m ""과 같다

Git과 경쟁업체

<https://bitbucket.org/>

<https://www.sourcetreeapp.com/> - 무료 다운로드하기

<https://git-fork.com/> - 유료



다 지우고 **Sourcetree** 로 다시 연습

Clone

Cloning is even easier if you set up a remote account

탐색

저장소 종류:  Git 저장소입니다

탐색

Local Folder:

> 고급 옵션

클론

Sourcetree

- ☒ Install
- ☒ Registration
- ☒ 도구 설치
- ☐ Preferences

Preferences

Before we finish, take a moment to configure these settings.

다음

 SSH 키를 불러오시겠습니까?

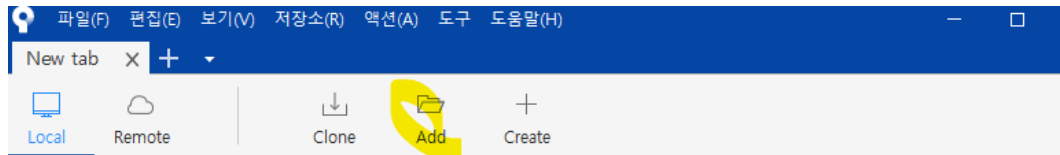
×



불러올 SSH 키를 하나 가지고 계십니까? 없으면 '아니오'를 누르고 나중에 하나 만드세요.

예

아니오

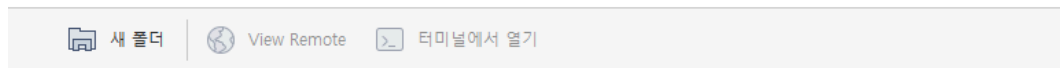


로컬 저장소

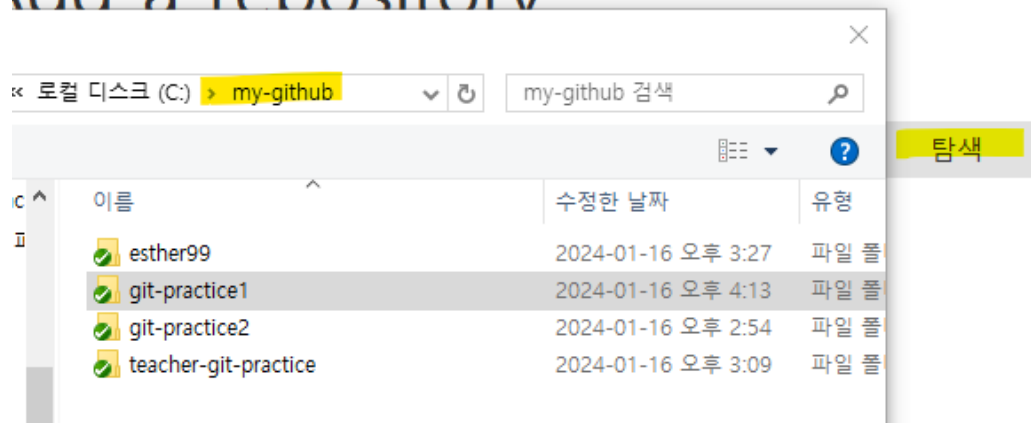
검색



시작하려면 새 저장소를 추가하거나 이 영역으로 저장소 폴더를 끌어다 놓으세요.



Add a repository



1월 17일 수요일

AM

branch - 나뉘어가지 > 비유

git branch

❖ git에서 main이 아닌 master 로 바꿔주기

The screenshot shows the GitHub repository settings for 'git-practice1'. The 'General' tab is selected. Under 'Code and automation', the 'Default branch' is set to 'main'. A modal dialog titled 'Rename this branch' is open, showing the process of renaming 'develop' to 'master'. The dialog includes a confirmation checkmark and a note that most projects name the default branch 'main'. A green button at the bottom of the dialog says 'Rename branch'.

Repository name: git-practice1

Repository type: ☐ Template repository

Require contributors to sign off on web-based commits: ☐ Require contributors to sign off on web-based commits

Default branch: main

Rename this branch

Rename develop to:

master

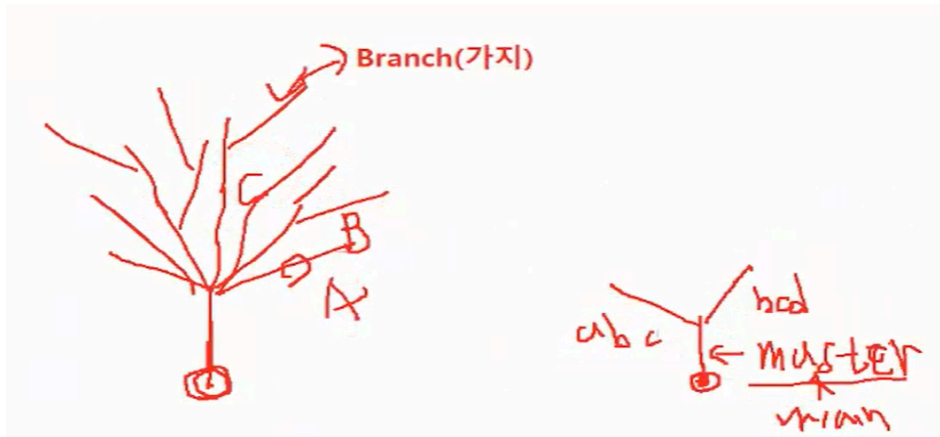
Most projects name the default branch main

Renaming this branch will not update your members' local environments.

Learn more

Rename branch

마스터 브랜치 읽기모드



마스터와 브랜치

디스크 (C:) > my-github > practice1

practice1 검색

이름	수정한 날짜	유형	크기
.git	2024-01-16 오후 5:21	파일 폴더	
esther.txt	2024-01-16 오후 5:20	텍스트 문서	
hi.txt	2024-01-16 오후 5:17	텍스트 문서	
README.md	2024-01-16 오후 5:17	Markdown 원본 ...	
README2.md	2024-01-16 오후 5:17	Markdown 원본 ...	

MINGW64:/c/my-github/practice1

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/practice1 (main)
$ git branch
* main
```

github에서 **matser**로 바꿔주고나서 **git**에서 **git branch**로 넣어주면 **main**으로 나온다

깃	!=	깃허브
branch		remote branch
local		
main		master > 싱크 맞춰주기

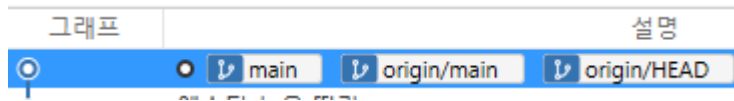
❖ **branch**란, 현재 로컬에 있는 모든 브랜치를 보여주는 명령어

git branch -r 입력 후, 마스터가 안나올경우 **git pull**을 하고 다시 **git branch-r** 입력

MINGW64:/c/my-github/practice1 (main)

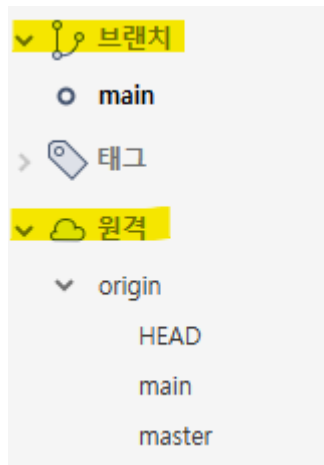
```
$ git branch -r
origin/HEAD -> origin/main
origin/main
origin/master
```

git branch -r 현재 원격에 있는 모든 브랜치를 보여주는 명령어



origin 최상의 루트. 원격에서 만든 최초의 루트, 고정되어있다

head 현재작업 브랜치의 위치, 현재의 위치를 나타내는 커서 **cursor**

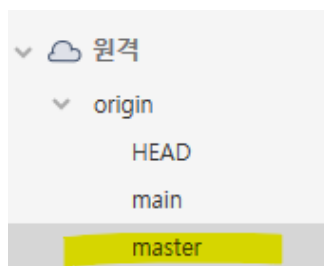


로컬브랜치 **Git**,
원격브랜치 **Github**

로컬브랜치에는 **master**가 없고 원격브랜치에서는 **main**과 **master**이 있다

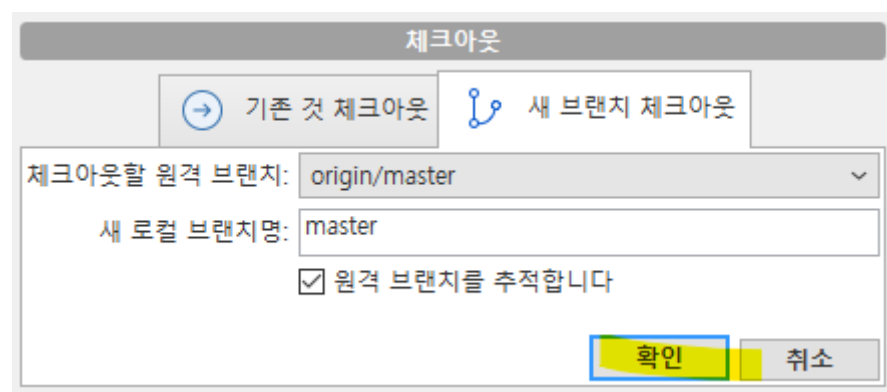
(main) > 헤더위치 이것을 바꾸려면?

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/practice1 (main)
$ .....
```

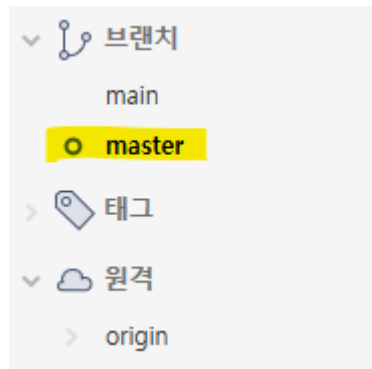


더블클릭

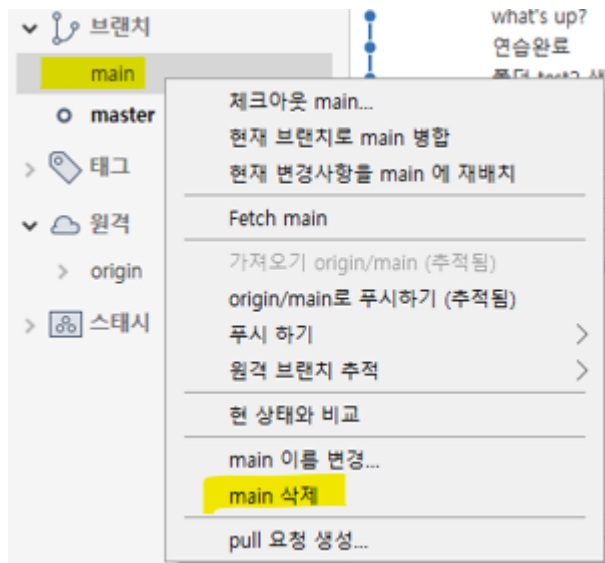
checkout 브랜치와 브랜치 사이를 이동



확인 누르면



마스터라는 브랜치가 생김



메인삭제

하고 다시 Git에서 **git branch** 를 입력하면 **main** 에서 **master**로 바뀐다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/practice1 (main)
$
```

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch
* master
```

Pull & Push 눌러주기

HTML CSS JS 통합코드 작성할 수 있는 곳: <https://codepen.io/pen/>

❖ **branch** 브랜치를 만들고 삭제하기

git branch -a 현재 로컬과 원격에 있는 모든 브랜치들을 모두 보여준다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/main
  remotes/origin/master
```

git branch 브랜치명 : 로컬 브랜치명을 만드는 명령어

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch test-branch1

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch
* master
  test-branch1
```

git branch -d 브랜치명 : 로컬 브랜치명을 삭제하는 명령어

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch -d test-branch1
Deleted branch test-branch1 (was faf252a).

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch
* master
```

git checkout 브랜치명: 브랜치명으로 작업영역을 이동/변경

git checkout -b 만들브랜치명: 만들 브랜치명이 만들어지고 바로 브랜치명으로 이동된다

```

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch test-branch1

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git checkout test-branch1
Switched to branch 'test-branch1'

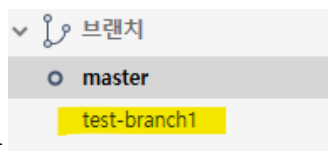
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (test-branch1)
$ git branch
  master
* test-branch1

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (test-branch1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch
* master
  test-branch1

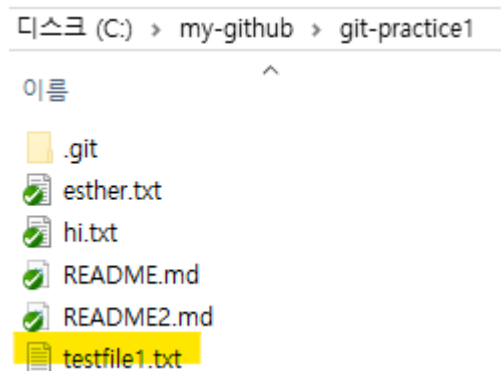
```

왜 이 작업을 하는가? **branch** 방을 나눠서 작업할 수 있다



sourcetree 에서 **F5** 해주면 보인다

❖ 테스트파일 한개 만들고 커밋하기



1. 테스트파일만들기
2. **git add testfile1.txt**
3. **git commit -m "테스트파일생성"**

Q. 만들어진 파일이름이 **testfile2** 후 **test-branch1**에서 커밋해주면 **testfile2**가 **testfile1**로 자동변경된다??

브랜치명을 깃에 작성 **git branch test-branch1**, **git checkout**으로 브랜치명을 바꿔주고 **pc**에서 **test**폴더생성 이것을 깃허브에 적용하려면 **git push**후 **git push --set-upstream origin test-branch1** 입력하면 깃허브에서도 똑같이 볼 수 있다

```

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git checkout test-branch1
Switched to branch 'test-branch1'

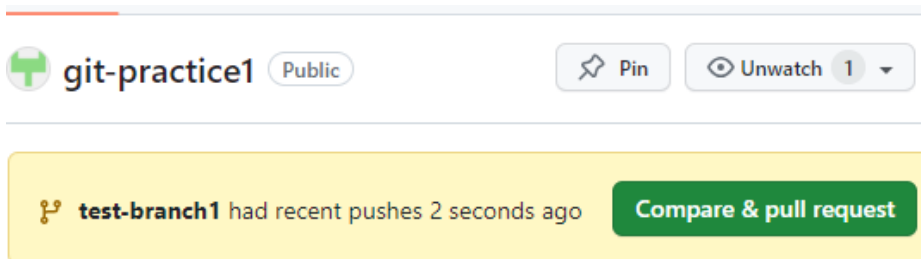
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (test-branch1)
$ git push
fatal: The current branch test-branch1 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin test-branch1

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (test-branch1)
$ git push --set-upstream origin test-branch1
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 239 bytes | 239.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'test-branch1' on GitHub by visiting:
remote:   https://github.com/icanbewhatever/git-practice1/pull/new/test-branch1
remote:
To https://github.com/icanbewhatever/git-practice1
 * [new branch]      test-branch1 -> test-branch1
branch 'test-branch1' set up to track 'origin/test-branch1'.

```



❖ 다시 브랜치 삭제해보기

마스터로 이동 **git checkout master - Q** 왜 마스터로 가야지? 현재 방에서 그 방을 삭제할 수는 없다 다른 방에서 삭제가능하다. 브랜치 = 방

```

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (test-branch1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

```

git branch -d 브랜치명 하면 **git branch**할때 **master**만 남는다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch -d test-branch1
warning: deleting branch 'test-branch1' that has been merged to
'refs/remotes/origin/test-branch1', but not yet merged to HEAD
Deleted branch test-branch1 (was 589c036).

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch
* master
```

다시 **git branch -r**을 하면 맨 마지막 **test-branch1**이 있다

git push origin -d test-branch1을 해주면 깃과 깃허브에서 완전히 삭제된다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git branch -r
origin/HEAD -> origin/master
origin/main
origin/master
origin/test-branch1

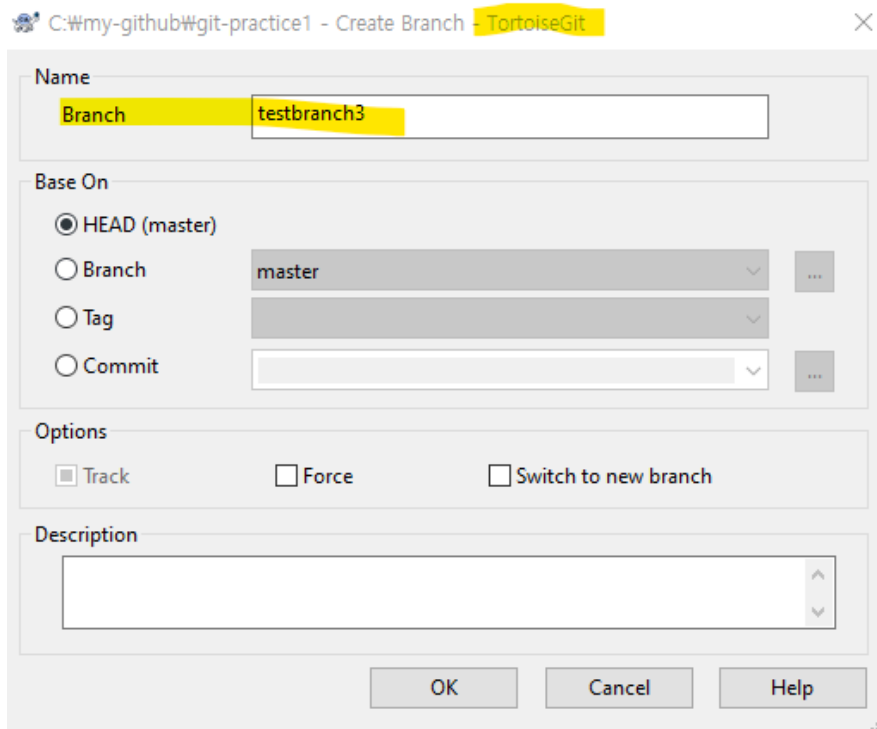
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git push origin -d test-branch1
To https://github.com/icanbewhatever/git-practice1
- [deleted]          test-branch1
```

❖ 연습하기

시나리오: 현재 **master**브랜치에 위치해 있다

1. testbranch2 브랜치 생성
→ `git branch testbranch2`
2. testbranch2 브랜치로 이동
→ `git checkout testbranch2`
3. testbranch2 브랜치에서 신규파일 생성
→ 파일 생성
4. 신규파일 add 및 commit
→ `git add testfile2.txt > git commit -m "두번째 브랜치"`
5. github에 testbranch2 브랜치 생성
→ `git push > git push --set-upstream origin test-branch2`
6. master 브랜치로 이동을 해서 신규파일이 없는 것을 확인
→ `git checkout master`
7. github의 testbranch2 브랜치 삭제
→ `git branch -d test-branch2`
8. 내 pc의 testbranch2 브랜치 삭제
→ `git branch -r / git push origin -d test-branch2`

토토리에서도 브랜치 만들 수 있다



❖ MERGE

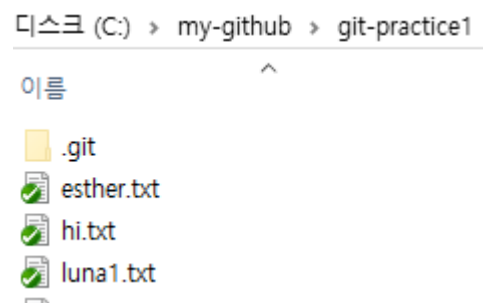
git merge 파일명 : 파일 합치기 / 파일이 있다가~ 없다가~

1. `git checkout -b` 브랜치명 : 브랜치를 새로만들고 바로 브랜치명으로 이동

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git checkout -b luna1
Switched to a new branch 'luna1'

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (luna1)
```

2. 파일만들고 토토로 `add & commit`



3. 만든 파일은 새 브랜치명을 **checkout** 할 때만 보인다. 그러므로 **checkout master**을 할 경우 파일이 사라졌다가 다시 **git checkout luna1** 을 입력하면 사라졌던 파일이 다시 보인다
4. 이 파일 **master**에서도 공유하고 싶다면, **git merge** 브랜치명을 입력할 경우 파일이 공유되면서 합쳐진다.
단, 공유파일이 있는 브랜치명에서 **merge**하지 않고 다른 브랜치명이나 **master**에서 실행한다.
즉, **git checkout master**을 했을때에도 파일을 볼 수 있다.

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (luna1)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git merge luna1
Updating b3aa502..d2289ba
Fast-forward
 luna2.txt | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 luna2.txt
```

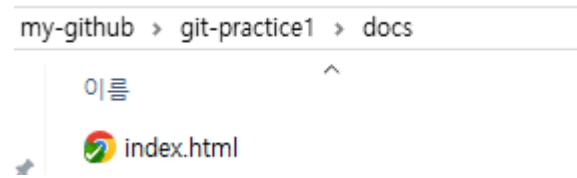
branch > add > commit > checkout > merge > branch > checkout 순환한다

마지막은 항상 **git push** 로 github와 연동하기

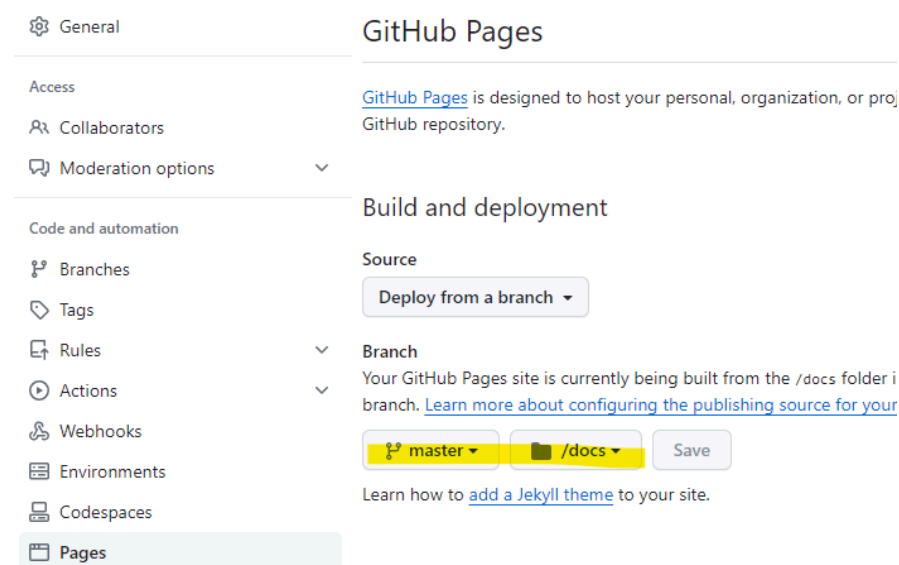
PM

❖ 폴더생성과 html

폴더 만들고 파일명 index.html 로 넣기 꼭, 폴더명은 docs 파일명은 index.html로 넣어야 한다

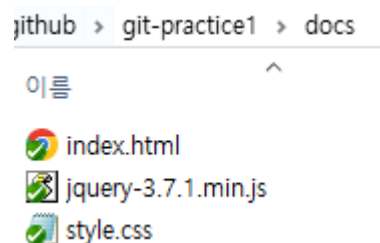


Github에 push 해주고 나서 setting에서 branch 아래처럼 맞춰주기



닉네임.github.io/레포지토리이름 을 주소창에 넣으면 나온다
<https://icanbewhatever.github.io/git-practice1/>

다른 파일도 넣어주기



<link rel="stylesheet" href="./style.css">

❖ git 충돌 (conflict)

branch를 b1, b2를 만들고 안에 똑같이 아래처럼 파일을 만들어 준다

```
echo "file1" >> unique1.txt
git add unique1.txt
git commit -m 'unique1.txt 파일 생성'
```


branch에서 b1을 merge하고 또 b2에서 merge를 하면 파일 이름과 안의 내용이 같기 때문에 충돌이 일어난다

```
human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master)
$ git merge b2
Auto-merging unique1.txt
CONFLICT (add/add): Merge conflict in unique1.txt
Automatic merge failed; fix conflicts and then commit the result.

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master|MERGING)
$ git add .

human-20@DESKTOP-85ADT1V MINGW64 /c/my-github/git-practice1 (master|MERGING)
$ git commit -m 'as'
[master 89c9935] as
```

충돌이 일어난 파일엔 경고표시가 나면 파일을 클릭해 안의 내용을 바꿔주고 다시 파일 add 와 commit을 해주면 오류는 사라진다.

 unique1.txt

❖ REMOTE MERGE 원격머지 (LOCAL MERGE는 잘 안쓴다)

❖ GIT & GITHUB 정리

linux ls명령어: 파일리스트 보기(-al -> 숨김파일 및 폴더까지 show)

linux cd명령어: 폴더 이동

git vs github

git: 소스 버전 관리 시스템(프로그램)

ex) 1일차: html, css, javascript 코드 작성

2일차: java작성 코드 작성

3일차: sql작성

github: git을 이용해서 클라우드 관리할 수 있게 해주는 사이트

git config --global user.email teacher@naver.com -> git에 내가 누구인지 세팅1

git config --global user.name teacher -> git에 내가 누구인지 세팅2

git init -> 로컬pc에 git설정 세팅

git add 파일명1 -> 새로 생성된 파일명1을 commit할 수 있도록 Staged Area에 등록

git add 파일명1 파일명2 -> 새로 생성된 파일명1, 파일명2 파일을 commit할 수 있도록 Staged Area에 등록하는 명령어

git add . -> 현재 디렉토리(폴더)의 모든 새로 생성된 파일을 commit할 수 있도록 Staged Area에 등록하는 명령어

git commit -m '메시지 내용' -> git 로컬에 반영(스냅샷, 해시 생성)

git remote add origin 원격저장지주소repostory: 원격 저장소(github 등)을 등록

git push -> 로컬 작업한 commit의 내용을 원격저장소(github)에 저장

git pull: 원격저장소에 있는 내용들을 전부 로컬git에 저장(+ local에 있는 내용들이랑 merge)

git branch: 현재 로컬에 있는 모든 브랜치를 보여주는 명령어

git branch -r: 현재 원격에 있는 모든 브랜치를 보여주는 명령어

git branch -a: 현재 로컬과 원격에 있는 모든 브랜치를 보여주는 명령어

git branch 브랜치명: 로컬 브랜치명을 만드는 명령어

git branch -d 브랜치명: 로컬 브랜치명을 삭제하는 명령어

* 원격 브랜치만 삭제 -> git push origin -d 브랜치명

git checkout 브랜치명: 브랜치명으로 작업영역을 변경

git checkout -b 만들브랜치명: 만들브랜치명이 만들어지고 바로 만들브랜치명으로 checkout해줌

origin: 원격에서 사용하는 최상위 원격 브랜치명

HEAD: 현재 작업 브랜치의 위치

checkout: 브랜치와 브랜치 사이를 이동

git 충돌(conflict) 해결: 서로 다른 브랜치에서 같은 파일 혹은 같은 줄의 있는 내용들을 바꿀 때 주로 발생.

git pull request(remote merge): github에서 remote 브랜치들을 merge해 주는 기능

webpage url -> 닉네임.github.io/레포지토리이름

<https://heropcode.github.io/GitHub-Responsive/img/logo.svg>