

(참고한 영상 from 알파한 코딩사전)

혼자 작업할 때 (설치 및 기본사용) - <https://www.youtube.com/watch?v=FXDjmsiv8fl>

공동 작업할 때 (github 연동 및 pull, push) - <https://www.youtube.com/watch?v=GaKjTjwckQo>

(이전에 공유한 재생목록 '종설관련'에도 추가되어 있습니다.)

([https://www.youtube.com/playlist?list=PLcZGvpZukJkTBYxTDsCgXv\\_ilubd0KyxB](https://www.youtube.com/playlist?list=PLcZGvpZukJkTBYxTDsCgXv_ilubd0KyxB))

(git 설치 참고)

<https://wonderbout.tistory.com/64>

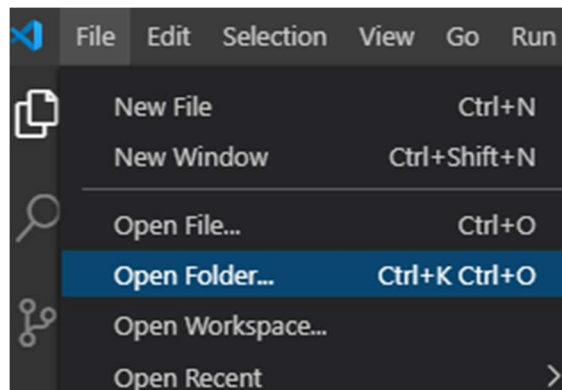
(git 사용법 참고)

<https://backlog.com/git-tutorial/kr/>

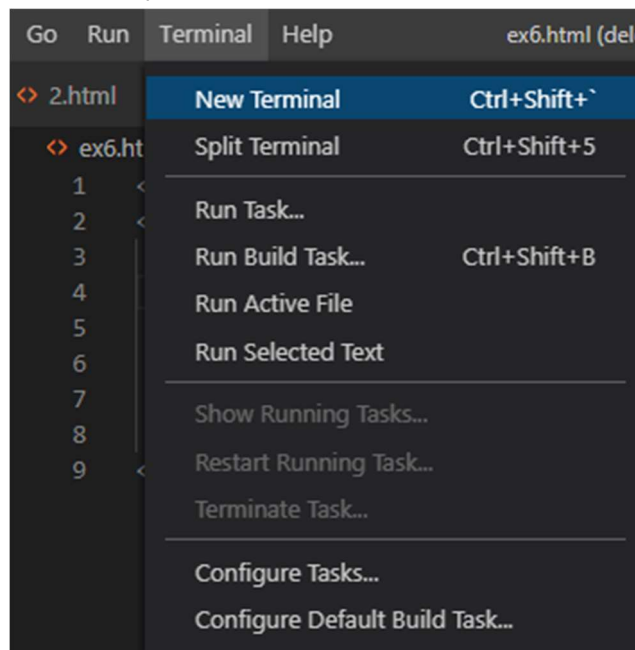
자주 사용하게 될 명령어는 **형광 펜**으로 칠해놨습니다.

#### \* git 기본 - 등록, log, commit, add, reset, revert

- 상단의 File - Open - Open Folder를 통해 작업할 프로젝트 폴더 open.



- 폴더를 연 후, Ctrl + `를 눌러 터미널 open. (또는, 상단의 Terminal - New Terminal 클릭)



(또는, cd 명령어를 통해 해당 디렉터리로 이동해도 무방)

- CLI에 **git init** 입력.

```
PS C:\Users\skdb\Desktop\web> git init
Initialized empty Git repository in C:/Users/skdb/Desktop/web/.git/
```

→ 이제부터 현재 작업중인 디렉터리는 git 관리를 받도록 하는 작업입니다.

- `git config --global user.name "사용자 이름"` & `git config --global user.email "이메일 주소"` 입력

```
PS C:\Users\skdbs\Desktop\web> git config --global user.name "skdbsxir"
PS C:\Users\skdbs\Desktop\web> git config --global user.email "skdbsxir@naver.com"
```

→ 처음 git 사용 시 파일을 push, commit 등의 작업을 할 때 누가 했는지 명시하기 위한 작업입니다.  
(한번 설정하고 나면 다음에 다시 설정할 필요가 없습니다.)

```
PS C:\Users\skdbs\Desktop\web> git config user.name
skdbsxir
PS C:\Users\skdbs\Desktop\web> git config user.email
skdbsxir@naver.com
```

→ 다음의 명령으로 사용자의 정보를 확인할 수 있습니다.

- `git status` 입력

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    1.html
    1.png
    2.html
    3.html
    box.html
    colors.js
    css
    ex1.html
    ex2.html
    fetch.html
    hash.html
    html
    index.html
    javascript
    list
    welcome

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

→ 현재 git이 추적하고 있는 파일들이 무엇인지 추적 상태를 확인하는 명령어입니다.

→ 현재는 아무런 파일이 commit 되지 않았기에, 다음과 같이 Untracked files에 현재 디렉터리에 있는 파일들이 전부 포함됩니다.

- `git add -A` (or `filename` (ex. hash.html)) 입력

> -A로 하면 해당 디렉터리의 모든 파일이, filename으로 하면 디렉터리의 해당 파일만이 git이 track할 목록에 추가됩니다.

```
PS C:\Users\skdbs\Desktop\web> git add -A
```

- 그 후 **git status**를 다시 입력해보면 git이 해당 파일들을 track하고 있음을 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   1.html
    new file:   1.png
    new file:   2.html
    new file:   3.html
    new file:   box.html
    new file:   colors.js
    new file:   css
    new file:   ex1.html
    new file:   ex2.html
    new file:   fetch.html
    new file:   hash.html
    new file:   html
    new file:   index.html
    new file:   javascript
    new file:   list
    new file:   welcome
```

- add된 내역을 확인했으면 **git commit -m "남길 메시지"**를 입력.

```
PS C:\Users\skdbs\Desktop\web> git commit -m "웹 실습 첫번째 커밋"
[master (root-commit) 4d0aa32] 웹 실습 첫번째 커밋
16 files changed, 408 insertions(+)
create mode 100644 1.html
create mode 100644 1.png
create mode 100644 2.html
create mode 100644 3.html
create mode 100644 box.html
create mode 100644 colors.js
create mode 100644 css
create mode 100644 ex1.html
create mode 100644 ex2.html
create mode 100644 fetch.html
create mode 100644 hash.html
create mode 100644 html
create mode 100644 index.html
create mode 100644 javascript
create mode 100644 list
create mode 100644 welcome
```

→ "남길 메시지" 부분에 현재 commit하는 파일들이 어떤 파일인지 설명해 줄 내용을 적어줍니다.

- **git status**를 통해 commit이 제대로 이루어졌음을 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master
nothing to commit, working tree clean
```

→ 현재 모든 파일들이 commit되었고, 수정된 사항이 없으므로 commit할 내용이 없다는 문구가 뜹니다.  
(초기 설정을 하면 master branch에서 작업이 이루어집니다. branch에 대한 설명은 뒤쪽에서 하겠습니다.)

- 파일 하나를 생성하고(저는 ex3.html로 했습니다), save한 다음 **git status**를 입력

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    ex3.html

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

→ 새로운 파일이 감지되었는데, git이 해당 파일을 추적하고 있지 못하고 있으며, git add를 통해 track할 목록에 추가해달라 하고 있습니다.

→ git add -A를 하고 commit을 해줍니다.

```
PS C:\Users\skdbs\Desktop\web> git add -A
PS C:\Users\skdbs\Desktop\web> git commit -m "두번째 커밋 연습"
[master 8428b97] 두번째 커밋 연습
1 file changed, 9 insertions(+)
create mode 100644 ex3.html
```

→ 해당 변경사항이 적용된 내용이 제대로 commit됩니다.

- **git log** 입력

```
PS C:\Users\skdbs\Desktop\web> git log
commit 8428b973e603233343e36339ead2aa8b12362ef3 (HEAD -> master)
Author: skdbsxir <skdbsxir@naver.com>
Date: Mon Jun 1 18:24:05 2020 +0900

    두번째 커밋 연습

commit 4d0aa32bbe2128e43ca074f39a09f9b478952c37
Author: skdbsxir <skdbsxir@naver.com>
Date: Mon Jun 1 18:21:13 2020 +0900

    웹 실습 첫번째 커밋
```

→ 아래에서 위 방향으로 commit된 내용의 이력을 볼 수 있습니다.

- 기존에 있는 파일의 내용을 수정하고, **git status**를 입력해 확인해봅시다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   ex3.html

no changes added to commit (use "git add" and/or "git commit -a")
```

→ ex3.html 파일이 수정되었지만, 마찬가지로 git이 해당 파일을 추적하고 있지 못하고 있으며, git add를 통해 track할 목록에 추가해달라 하고 있습니다.

→ 마찬가지로 git add -A, git commit을 해줍니다.

```
PS C:\Users\skdbs\Desktop\web> git commit -m "세번째 커밋 수정 후"
[master ea51acc] 세번째 커밋 수정 후
1 file changed, 1 insertion(+), 1 deletion(-)
```

→ git log 또는 git status를 통해 제대로 commit이 되었는지 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\skdbs\Desktop\web> git log
commit ea51acc7d7da41e99332d15666dcb2444c145ab8 (HEAD -> master)
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:25:38 2020 +0900

    세번째 커밋 수정 후

commit 8428b973e60323343e36339ead2aa8b12362ef3
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:24:05 2020 +0900

    두번째 커밋 연습

PS C:\Users\skdbs\Desktop\web> git reset 8428b9 --hard
HEAD is now at 8428b97 두번째 커밋 연습
PS C:\Users\skdbs\Desktop\web> git log
commit 8428b973e60323343e36339ead2aa8b12362ef3 (HEAD -> master)
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:24:05 2020 +0900

    두번째 커밋 연습

commit 4d0aa32bbe2128e43ca074f39a09f9b478952c37
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:21:13 2020 +0900

    웹 실습 첫번째 커밋
```

- 현재 프로젝트의 작업 내용을 전부 지우고, 이전 작업 시점으로 돌아가고자 할 땐 **git reset** 명령을 이용합니다.

> **git reset 'commit의 일련번호 맨 앞 6자리' --hard**

(일련번호 6자리는 git log를 통해 확인할 수 있습니다.)

```
commit 8428b973e60323343e36339ead2aa8b12362ef3
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:24:05 2020 +0900

    두번째 커밋 연습
```



```
commit 8428b973e603233343e36339ead2aa8b12362ef3
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:24:05 2020 +0900

두번째 커밋 연습
PS C:\Users\skdb\Desktop\web> git reset 8428b9 --hard
HEAD is now at 8428b97 두번째 커밋 연습
```

→ 그러면 git의 HEAD 포인터가 돌아가고자 하는 해당 commit을 가리키고 있다 알려줍니다.

→ log를 통해 살펴보면 이전에 commit한 내역은 사라지고, 돌아온 commit까지만이 남아있음을 알 수 있습니다.

```
PS C:\Users\skdb\Desktop\web> git log
commit 8428b973e603233343e36339ead2aa8b12362ef3 (HEAD -> master)
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:24:05 2020 +0900

두번째 커밋 연습

commit 4d0aa32bbe2128e43ca074f39a09f9b478952c37
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:21:13 2020 +0900

웹 실습 첫번째 커밋
```

→ git reset 8428b9 --hard 수행 후 master branch의 최신은 해당 commit임을 알 수 있습니다.

**!reset '~' --hard 를 사용하면, 물리적으로 링크 연결을 끊는 것이기에, 직전에 작업했던 commit을 되돌릴 수 없습니다!**

- **git revert** 명령을 이용하면 현재 commit을 취소한 상태에서 이전 commit상태로 돌아갈 수 있습니다.

> **git revert 'commit의 일련번호 맨 앞 6자리'**

**!이때 일련번호 6자리는 취소하고자 하는 commit의 맨 앞 6자리 입니다!**

```
PS C:\Users\skdb\Desktop\web> git log
commit 8372d6cdedf657bb01f3e7cea0b2b6572bc1ef4d (HEAD -> master)
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:28:35 2020 +0900

다섯번째 연습용

commit 42c7f9517913adbd156dcec7b3fdbc4f365c4590
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:27:40 2020 +0900

네번째 연습용

commit 8428b973e603233343e36339ead2aa8b12362ef3
Author: skdbxsir <skdbxsir@naver.com>
Date: Mon Jun 1 18:24:05 2020 +0900

두번째 커밋 연습
PS C:\Users\skdb\Desktop\web> git revert 8372d6
hint: Waiting for your editor to close the file...
```

→ 이런 경우 확인 창이 뜨는데, CLI에 뜰 수도 있고 위 창(파일 open한 창)에 뜰 수도 있습니다.

```
1 COMMIT_EDITMSG
1 Revert "ADD MODIFY DELETE"
2
3 This reverts commit 68f0c5ba93513be955e8ad70bca43022d3dcc0cc.
4
5 # Please enter the commit message for your changes. Lines starting
6 # with '#' will be ignored, and an empty message aborts the commit.
7 #
8 # On branch master
9 # Changes to be committed:
10 #   new file:   cat
11
12 :wq
```

→ 그대로 저장한다는 명령어 **:wq**를 입력 후 엔터를 눌러주면, commit이 완료됩니다.

```
PS C:\Users\skdbs\Desktop\web> git revert 8372d6
hint: Waiting for your editor to close the file...
[master 7f14cb2] Revert "다섯번째 연습용"
1 file changed, 1 deletion(-)
```

→ log를 통해 확인해보면, reset때처럼 이전 commit이 사라진 것이 아니라 새로운 commit으로 보존되고 있음을 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git log
commit 7f14cb247d319b5a48b81428b6fa4e7984c0e3a1 (HEAD -> master)
Author: skdbsxir <skdbsxir@naver.com>
Date: Mon Jun 1 18:29:31 2020 +0900

    Revert "다섯번째 연습용"

This reverts commit 8372d6cdedf657bb01f3e7cea0b2b6572bc1ef4d.

:wq

commit 8372d6cdedf657bb01f3e7cea0b2b6572bc1ef4d
```

→ 현재 작업중인 곳은 8372d6... 입니다.

## \* git 기본 - branch

- 현재 원본은 그대로 내버려 두고, 같은 내용으로 다른 시도를 하고자 할 때 생성/사용하는 것이 branch입니다. (영상을 보시면 **branch == 평행우주**라고 설명해줍니다)

- **git branch 'branch 이름'**을 통해 branch 생성

```
PS C:\Users\skdb\Desktop\web> git branch example-bran
```

→ 생성한 branch는 **git branch**로 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git branch
example-bran
* master
```

→ 현재 초록색으로 표시되어 있고, 앞에 \* 가 붙은 branch가 현재 작업중인 branch임을 의미합니다.

- **git checkout '이동하고자 하는 branch 이름'**을 통해 작업 위치를 해당 branch로 이동할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git checkout example-bran
Switched to branch 'example-bran'
PS C:\Users\skdbs\Desktop\web> git checkout master
Switched to branch 'master'
```

- 현재 작업하고 있었던 파일들을 그대로 불러오기 위해선, **git add -A** 명령어를 그대로 쳐주면 됩니다.

```
PS C:\Users\skdbs\Desktop\web> git add -A
```

- 해당 branch에 있는 상태에서, 현재 작업중인 파일 (ex3.html)의 내용을 변경하고 **git status**를 입력해봅니다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch example-bran
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   ex3.html
```

- commit을 하면 현재 위치하고 있는 branch에 내용들이 commit되는 것을 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git status
On branch example-bran
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   ex3.html

PS C:\Users\skdbs\Desktop\web> git commit -m "branch practice 1"
[example-bran a05e322] branch practice 1
1 file changed, 12 insertions(+)
```

- 다른 branch에서 변경한 사항을 현재 branch에 합치고 싶을 땐, **git merge**를 사용합니다.

> (현재 위치한 branch에서) **git merge '현재 branch에 병합하고자 하는 branch의 이름'**

> (현재 master branch에 위치, example-bran의 내용을 master에 병합)

→ **git merge example-bran**

```
PS C:\Users\skdbs\Desktop\web> git add -A
PS C:\Users\skdbs\Desktop\web> git commit -m "before merging"
[master 6f65e8a] before merging
1 file changed, 1 insertion(+)
```

```
PS C:\Users\skdbs\Desktop\web> git log
commit 6f65e8a09f1dd018e1a55cbe45a17f51e97afa27 (HEAD -> master)
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:37:01 2020 +0900

    before merging

commit 7f14cb247d319b5a48b81428b6fa4e7984c0e3a1
Author: skdbsxir <skdbsxir@naver.com>
Date:   Mon Jun 1 18:29:31 2020 +0900

    Revert "다섯번째 연습용"
```



```
PS C:\Users\skdbs\Desktop\web> git merge example-bran
Auto-merging ex3.html
CONFLICT (content): Merge conflict in ex3.html
Automatic merge failed; fix conflicts and then commit the result.
```

(conflict가 발생한 상황입니다)

**!같은 파일의 같은 부분(라인)을 수정하면 conflict가 발생합니다!**

conflict가 발생하면, git이 자동 merge를 거부하고 해당 conflict를 해결하고 와야 합니다.  
(해당 부분을 캡처 해 둔 것이 없어 영상의 예시로 대체합니다)

```
1 name: mong
   현재 변경 사항 수락 | 수신 변경 사항 수락 | 두 변경 사항 모두 수락 | 변경 사항 비교
2 <<<<<< HEAD (현재 변경 사항)
3 bark: warl warl
4 =====
5 bark: wang wang
6 >>>>>> bark-wang (수신 변경 사항)
```

```
dog
1 name: mong
2 bark: warl warl
3 =====
4 bark: wang wang
5 >>>>>> bark-wang
6
```

```
1 name: mong
2 bark: warl warl
3
```

- 수정하고자 하는 내용을 지우고, **git add -A**를 입력한 다음 **git commit** 만을 입력합니다.
- 위의 revert 명령을 했을 때 처럼 창이 나오고, :wq를 입력해줍니다.

```
PS C:\Users\skdbs\Desktop\web> git merge example-bran
Merge made by the 'recursive' strategy.
ex3.html | 1 +
1 file changed, 1 insertion(+)
```

```
PS C:\Users\skdbs\Desktop\web> git log
commit 545dedf744b7056b352101802ef4c4394c710331 (HEAD -> master)
Merge: 48b2a97 9dcbe6c
Author: skdbsxir <skdbsxir@naver.com>
Date: Mon Jun 1 18:53:44 2020 +0900

Merge branch 'example-bran'
```

(conflict 없이 제대로 merge가 이루어진 상태입니다.)

- **git log --graph --all --decorate**를 입력하면 현재까지의 진행 상황을 그래프로 그려서 출력해줍니다.

```
PS C:\Users\skdbs\Desktop\web> git log --graph --all --decorate
* commit 545dedf744b7056b352101802ef4c4394c710331 (HEAD -> master)
| Merge: 48b2a97 9dcbe6c
| Author: skdbsxir <skdbsxir@naver.com>
| Date: Mon Jun 1 18:53:44 2020 +0900
|
| Merge branch 'example-bran'
|
| * commit 9dcbe6cbd66df40bbbd06ad42f137808d6d63429 (example-bran)
| | Author: skdbsxir <skdbsxir@naver.com>
| | Date: Mon Jun 1 18:50:51 2020 +0900
| |
| | example commit 2'
|
| * commit 48b2a972d98f124e22ea9c0805129fca3c5be4a5
| | Author: skdbsxir <skdbsxir@naver.com>
| | Date: Mon Jun 1 18:53:16 2020 +0900
```



- git rebase 를 통해 이런 branch의 갈래를 간략하게 정리할 수 있습니다.

(가장 쉬운 Git 강좌 - (상) 혼자작업편 영상의 13:17 부분을 참고해주세요 <https://youtu.be/FXDjmsiv8fl?t=797>)

### \* git & github 이용 (본인이 생성 및 관리)

- github에서 Repository를 생성한 다음, git CLI에서 해당 내용을 복사한 다음 실행합니다.

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

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/skdbsxir/WEbprac.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 "# WEbprac" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/skdbsxir/WEbprac.git
git push -u origin master
```

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

```
git remote add origin https://github.com/skdbsxir/WEbprac.git
git push -u origin master
```

#### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

```
PS C:\Users\skdb\ Desktop\web> git remote add origin https://github.com/skdbsxir/WEbPractice.git
```

→ origin은 원하는 다른 이름으로 바꾸어도 괜찮습니다. default로 설정된 이름입니다.

- 현재 commit한 내용들을 github의 해당 repository에 추가해줍니다.

> **git push -u origin(또는 설정한 이름) master(또는 올리고자 하는 branch이름)**

```
PS C:\Users\skdb\ Desktop\web> git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 4 threads
Compressing objects: 100% (43/43), done.
Writing objects: 100% (45/45), 26.73 KiB | 4.46 MiB/s, done.
Total 45 (delta 21), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (21/21), done.
To https://github.com/skdbsxir/WebPractice.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

→ 내용이 master branch에 push가 완료되었음을 알려줍니다.

**!이때 github의 계정, 비밀번호를 입력하고 접근권한을 허용해줘야 push가 진행됩니다!**

- git remote를 입력하면 현재 추가한 원격 Repository의 이름이 나옵니다.

```
PS C:\Users\skdbs\Desktop\web> git remote  
origin
```

(github repository 페이지를 새로 고침 하면 내용이 추가된 것을 확인할 수 있습니다.)

- 파일을 하나 생성하고(저는 ex4.html로 했습니다), 원격 Repository에 추가할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git add -A  
PS C:\Users\skdbs\Desktop\web> git commit -m "ex4 added"  
[master 8c5a69f] ex4 added  
1 file changed, 9 insertions(+)  
create mode 100644 ex4.html  
PS C:\Users\skdbs\Desktop\web> git push origin master  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/skdbsxir/WebPractice.git  
545dedf..8c5a69f master -> master
```

- 현재 작업중인 디렉터리에 .gitignore 파일을 생성해서 push에 제외 할 파일을 지정할 수 있습니다.  
> .gitignore 파일에 제외 할 파일의 이름을 추가해주면 됩니다.

```
> OPEN EDITORS  
v WEB  
  .gitignore  
    1 tohide.html
```

```
PS C:\Users\skdbs\Desktop\web> git status  
On branch master  
Your branch is up to date with 'origin/master'.  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  .gitignore  
  
PS C:\Users\skdbs\Desktop\web> git add -A  
PS C:\Users\skdbs\Desktop\web> git commit -m "add .gitignore"  
[master fe7871c] add .gitignore  
1 file changed, 1 insertion(+)  
create mode 100644 .gitignore
```

- 한번 remote가 설정되고, push를 진행하면 추 후에 git push만을 이용해 동일 위치로의 push가 가능합니다.

```
PS C:\Users\skdbs\Desktop\web> git push  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 280 bytes | 280.00 KiB/s, done.  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/skdbsxir/WebPractice.git  
8c5a69f..fe7871c master -> master
```

- Repository의 Settings – Manage access에 가서 해당 Repository에 파일을 수정 및 업로드 할 수 있는 사람을 지정할 수 있습니다.

The screenshot shows the GitHub repository settings page. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, 'Manage access' is highlighted under the 'Options' section. The main content area is titled 'Who has access' and shows two access types: 'PUBLIC REPOSITORY' (This repository is public and visible to anyone) and 'DIRECT ACCESS' (1 has access to this repository. 1 collaborator). Below this, the 'Manage access' section is visible, featuring a search bar 'Find a collaborator...' and a list of collaborators. One collaborator, 'T-Hiro12', is listed with a checkbox and a trash icon. A green button 'Invite a collaborator' is located in the top right corner of the 'Manage access' section.

- 다른 컴퓨터에서 자신이 작업한 Repository를 받아오거나, 다른 사람이 작업한 Repository를 받아오려면 **git clone 'Repository 주소'**를 입력해 받아올 수 있습니다.

```
PS C:\Users\junse\OneDrive\바탕 화면\my-practice> git clone https://github.com/channel-yalco/github-practice.git
Cloning into 'github-practice'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 10 (delta 0), reused 10 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.
PS C:\Users\junse\OneDrive\바탕 화면\my-practice> |
```

> **git log**를 입력하면 진행되었던 commit 내역을 그대로 확인할 수 있습니다.



- 다른 이가 Repository에 올린 내역을 일일이 확인하는 것은 번거로우므로, **git fetch**와 **git status**을 통해 확인할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 319 bytes | 7.00 KiB/s, done.
From https://github.com/skdbsxir/WebPractice
   fe7871c..c584323 master    -> origin/master
PS C:\Users\skdbs\Desktop\web> git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)

nothing to commit, working tree clean
```

→ 현재 원격으로 관리되는 origin/master에 비해 commit 하나가 뒤쳐져 있다고 알려줍니다.

→ 갱신된 내용을 다운받아줘야 합니다.

→ **git pull '원격 이름' 'branch 이름'**을 입력해 받아올 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git pull origin master
From https://github.com/skdbsxir/WebPractice
 * branch            master    -> FETCH_HEAD
Updating fe7871c..c584323
Fast-forward
 ex5.html | 9 ++++++++
 1 file changed, 9 insertions(+)
 create mode 100644 ex5.html
```

(다른 컴퓨터에서 작업하고 업로드한 ex5.html 파일을 받아온 것을 확인할 수 있습니다)

**!다른 이가 작업한 내역을 먼저 받아 놔야 괜한 작업을 하거나 conflict가 발생하는 일을 방지할 수 있습니다!**

→ 작업을 시작하기 전 **fetch, status**로 확인한 다음 **pull**을 통해 먼저 받아오는 것을 습관화 하면 좋습니다.

- branch를 생성하고, 해당 branch로 바로 이동하려면 **git checkout -b 'branch 이름'** 명령을 이용합니다.

```
PS C:\Users\skdbs\Desktop\web> git checkout -b desktop-prac
Switched to a new branch 'desktop-prac'
```

```
PS C:\Users\skdbs\Desktop\web> git branch
* desktop-prac
 example-bran
 master
```

> **git push '원격이름' 'branch이름'**을 통해 해당 branch에서 작업한 내역을 원격 관리중인 github repository에 업로드 할 수 있습니다.

```
PS C:\Users\skdbs\Desktop\web> git push origin desktop-prac
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 354 bytes | 354.00 KiB/s, done.
Total 3 (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 'desktop-prac' on GitHub by visiting:
remote:   https://github.com/skdbsxir/WebPractice/pull/new/desktop-prac
remote:
To https://github.com/skdbsxir/WebPractice.git
 * [new branch]    desktop-prac -> desktop-prac
```

[Code](#)
[Issues 0](#)
[Pull requests 2](#)
[Actions](#)
[Projects 0](#)
[Wiki](#)
[Security 0](#)
[Insights](#)
[Settings](#)

영상 보고 실습진행 Edit

Manage topics

18 commits    3 branches    0 packages    0 releases    1 contributor

Branch: master New pull request

Switch branches/tags

Find or create a branch...

Branches    Tags

✓ master default

desktop-prac

notebook-prac

com/skdbsxir/WebPractice

Latest commit 2dcabd9 2 hours ago

add .gitignore	4 hours ago
웹 실습 첫번째 커밋	6 hours ago
웹 실습 첫번째 커밋	6 hours ago
웹 실습 첫번째 커밋	6 hours ago
웹 실습 첫번째 커밋	6 hours ago
웹 실습 첫번째 커밋	6 hours ago

(해당 branch가 원격 저장소의 branch에 추가된 것을 확인할 수 있습니다)

- 원격 저장소의 branch를 받아오기 위해선

1) **git fetch** 명령을 통해 원격 저장소의 최신 상태를 받아오고

```

hyeonmin ~/Desktop/github-practice master git fetch
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 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/channel-yalco/github-practice
* [new branch] my-idea -> origin/my-idea
  
```

2) **git branch -a** 를 통해 원격 저장소와 로컬 저장소에 있는 branch를 확인한 다음

```

hyeonmin ~/Desktop/github-practice master git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/my-idea
(END)
  
```

3) **git checkout -b '로컬에 생성할 branch이름' '원격저장소에서 가져올 branch 경로/이름'**을 입력합니다.

```

hyeonmin ~/Desktop/github-practice master git checkout -b my-idea origin/my-idea
Branch 'my-idea' set up to track remote branch 'my-idea' from 'origin'.
Switched to a new branch 'my-idea'
  
```

```

PS C:\Users\skdbs\Desktop\web> git checkout -b fromnotebook origin/notebook-prac
Switched to a new branch 'fromnotebook'
Branch 'fromnotebook' set up to track remote branch 'notebook-prac' from 'origin'.
PS C:\Users\skdbs\Desktop\web> git branch
desktop-prac
example-bran
* fromnotebook
master
  
```

(노트북에서 생성하고 업로드 한 'notebook-prac' branch의 내용을 새로운 로컬 branch 'fromnotebook'에 가져와 생성한 모습입니다)

**!같은 branch에서 같은 파일의 같은 부분을 원격으로 2개 이상의 컴퓨터에서 건드릴 경우에도 conflict가 발생합니다!**

- 원격 저장소의 branch를 삭제하고자 할 땐 `git push -d '원격이름' 'branch이름'`을 입력해주면 됩니다.  
(로컬의 branch를 없앨 땐 `git branch -D 'branch이름'`을 입력하면 됩니다)

```
PS C:\Users\skdbs\Desktop\web> git branch -d fromnotebook
warning: deleting branch 'fromnotebook' that has been merged to
'refs/remotes/origin/notebook-prac', but not yet merged to HEAD.
Deleted branch fromnotebook (was af9692d).
PS C:\Users\skdbs\Desktop\web> git push -d origin notebook-prac
To https://github.com/skdbsxir/WebPractice.git
- [deleted]          notebook-prac
```

#### \* 추가

```
warning: LF will be replaced by CRLF in css/ie/PIE.htc.
The file will have its original line endings in your working directory
```

이런 경고문구가 뜰 수 있는데, 이는 다른 OS환경에서 작업한 파일을 git으로 협업할 때 발생하는 문구라고 합니다. (<https://blog.jaeyoon.io/2018/01/git-crlf.html>)

- `core.autocrlf`를 커서 해결할 수 있습니다.

그러므로 윈도우 사용자의 경우 이러한 변환이 항상 실행되도록 다음과 같은 명령어를 입력한다. 물론 시스템 전체가 아닌 해당 프로젝트에만 적용하고 싶다면 `-global`을 빼주면 된다.

```
git config --global core.autocrlf true
```

리눅스나 맥을 사용하고 있는 경우, 조회할 때 LF를 CRLF를 변환하는 것은 원하지 않을 것이다. 따라서 뒤에 `input`이라는 명령어를 추가해줌으로써 단방향으로만 변환이 이루어지도록 설정한다.

```
git config --global core.autocrlf true input
```

혹은 이러한 변환 기능을 원하지 않고, 그냥 에러 메시지 끄고 알아서 작업하고 싶은 경우에는 아래 명령어로 경고메시지 기능인 `core.safecrlf`를 꺼주면 된다.

```
git config --global core.safecrlf false
```