

## • GitHub 가입하고 내 컴퓨터에 Git 설치하기

01) GitHub 사이트(<https://github.com>)에 접속해서 [Sign up]버튼을 클릭한다. Username, Email, Password를 입력하고 초록색 [Sign up for GitHub] 버튼을 클릭한다. 다른 GitHub 유저가 이미 사용하고 있는 이름은 Username으로 사용할 수 없고 Email 주소도 마찬가지이다. 기재한 Email 주소로 GitHub에서 확인 메일을 보낸다.

02) 인증 절차를 거쳐서 [Create an account] 버튼이 활성화되면 클릭한다.

03) 다음은 GitHub를 무료 버전으로 사용할지, 아니면 매월 비용을 지불하는 유료버전으로 사용할지 선택하는 페이지에서 기본값인 [free]를 선택하고 다른 옵션은 그대로 두고 가장 아래에 있는 [Continue] 버튼을 클릭한다.

04) 몇 가지 설문 사항에 체크하고(체크하지 않아도 된다.) 가장 아래에 있는 [Submit]을 클릭한다.

05) GitHub에서 메일 주소 검증을 위한 메일을 발송하면 메일함을 열어 [Verify email address]를 클릭하면 가입이 완료된다.

06) <https://git-scm.com/downloads>에서 운영체제에 맞는 Git을 다운로드 받은 후 설치한다.

07) Git bash를 실행한 후 아래와 같이 '\$'표시 옆에 'git' 이라고 입력 후 리턴키를 누른다.

## • 로컬저장소 만들기

로컬 저장소는 실제로 Git을 통해 버전 관리가 이루어질 내 컴퓨터에 있는 폴더이다.

01) [c:/]-[home]-[tshirt] 폴더를 생성한다.

02) 위에서 생성한 폴더에 [readme.txt] 파일을 생성한 후 아래와 같이 내용을 입력 한다.

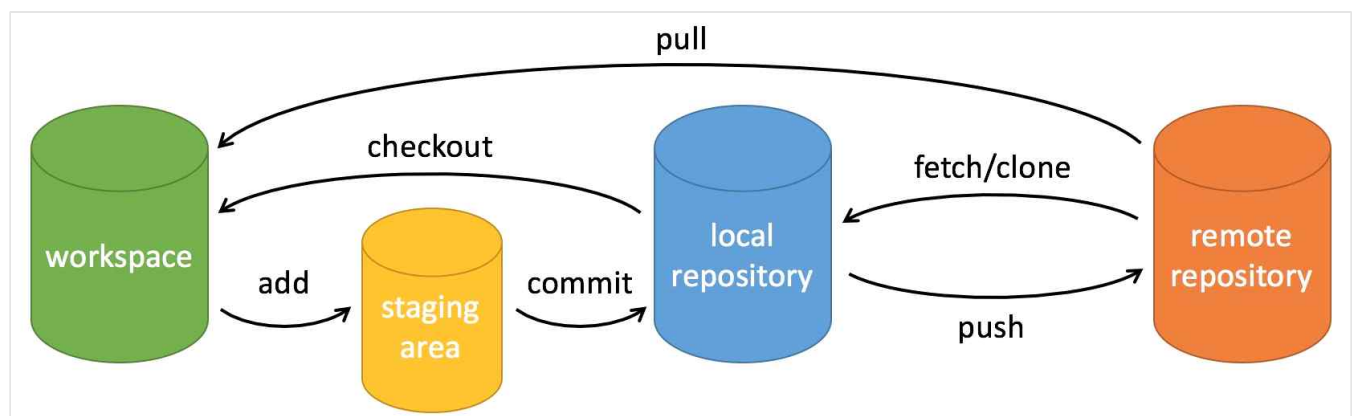


03) [tshirt] 폴더에서 마우스 오른쪽 버튼을 클릭하고 [Git Bash Here]를 클릭한다.

04) git init 명령을 실행하면 [tshirt] 폴더에는 [.git]라는 폴더가 자동으로 생성된다. [.git] 폴더에는 Git으로 생성한 버전들의 정보와 원격저장소 주소 등이 들어있다. [.git] 폴더를 로컬 저장소라고 부른다. (탐색기-> 보기 -> 숨긴 항목 체크)

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt
$ git init
Initialized empty Git repository in c:/home/tshirt/.git/
```

## • 첫 번째 커밋 만들기



위에서 생성했던 readme.txt 파일을 하나의 버전으로 만들어보겠다. Git에서는 이렇게 생성된 각 버전을 커밋이라고 부른다.

01) GitHub에서 실습을 진행할 예정이므로 GitHub 가입 시에 사용한 user.email과 user.name을 정확히 설정한다.

- --global 옵션은 모든 프로젝트에 적용된다. 프로젝트마다 다른 이름과 이메일 주소를 사용하고 싶으면 --global 옵션을 빼고 실행한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git config --global user.email "hd9536@hanmail.net"

jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git config --global user.name "helen-cho"
```

- 정보 삭제: \$ git config --global --unset user.email

02) 아래 명령어로 프로젝트의 모든 옵션을 살펴본다.

- 'q'를 눌러 빠져나올 수 있다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git config --list
```

03) 다음으로 커밋에 추가할 파일을 선택한다. 조금 전에 만들어 놓은 readme.txt 파일로 해 보겠다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git add readme.txt
```

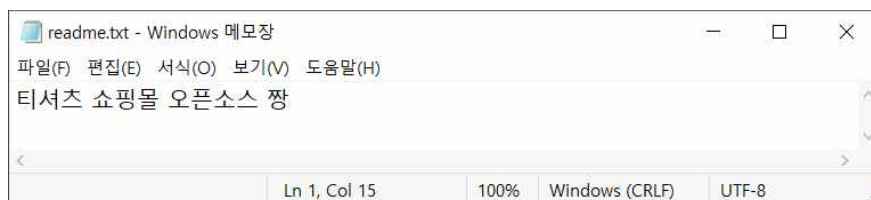
- 현재 폴더의 모든 파일 stage에 추가 : \$ git add .
- 지정 폴더를 stage에 추가 : \$ git add 폴더명
- 특정 파일 stage에서 삭제 : \$ git rm --cached readme.txt
- state에 모든 파일 삭제 : \$ git rm -r --cached .
- 파일 상태 확인하기 : \$ git status

04) 커밋에는 상세 설명을 적을 수 있다. 설명을 잘 적어놓으면 내가 이 파일을 왜 만들었는지, 왜 수정했는지 알 수 있고, 해당 버전을 찾아 그 버전으로 롤백 할 수 있다.

- [-m]은 'message'의 약자이다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git commit -m "사 이 트 설 명 추 가"
[master (root-commit) 014f721] 사 이 트 설 명 추 가
1 file changed, 1 insertion(+)
create mode 100644 readme.txt
```

05) readme.txt 파일을 아래와 같이 수정한다.



06) add 명령어로 readme.txt를 선택하고 '설명 업데이트'라는 설명을 붙여서 commit 명령어로 커밋을 만든다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git add readme.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git commit -m "설 명 업 데 이 트"
[master 1c79ec2] 설 명 업 데 이 트
1 file changed, 1 insertion(+), 1 deletion(-)
```

## • 다른 커밋으로 시간 여행하기

개발을 하다가 요구사항이 바뀌어서 이전 커밋부터 다시 개발하고 싶다면 Git을 사용해 이전 커밋으로 돌아갈 수 있다.

01) 먼저 log 명령어를 사용해 지금까지 만든 커밋을 확인한다. 두 개의 커밋을 확인 할 수 있다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git log
commit 1c79ec2e8147db9b6be553495f02762b422f1600 (HEAD -> master)
Author: helen-cho <hd9536@hanmail.net>
Date: Sun Jun 6 14:40:46 2021 +0900

    설명 업데이트

commit 014f72138b09725fb2318a6f285d51fbca018d7b
Author: helen-cho <hd9536@hanmail.net>
Date: Sun Jun 6 14:28:30 2021 +0900

    사이트 설명 추가
```

02) 첫 번째 커밋으로 되돌리려면 앞 7자리 또는 전체 커밋 아이디를 복사하고 checkout 명령어로 해당 커밋으로 코드를 되돌린다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git checkout 014f721
Note: switching to '014f721'.
```

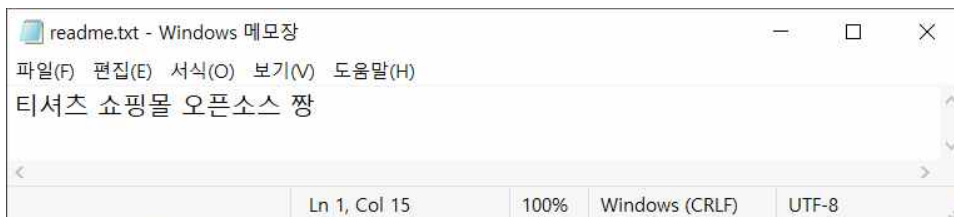
03) readme.txt 파일을 열어보면 '짱'이 사라진 첫 번째 커밋으로 돌아간 것을 확인할 수 있다.



04) 다시 체크아웃을 해서 최신 커밋인 두 번째 커밋으로 돌아가려면 커밋아이디를 사용해도 되지만 '-'를 적어도 된다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt ((014f721...))
$ git checkout -
Previous HEAD position was 014f721 사이트 설명 추가
Switched to branch 'master'
```

05) readme.txt 파일을 다시 열어보면 '짱'이 붙어 있는 두 번째 커밋으로 다시 돌아간 걸 확인할 수 있다.



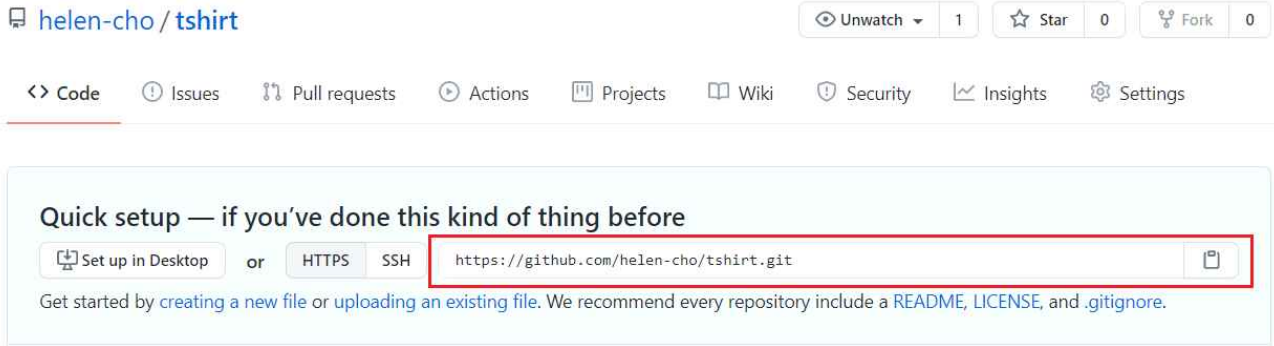
## • 원격저장소 만들기

이번에는 GitHub에 협업할 공간인 원격 저장소를 만든다. 쉽게 설명하는 GitHub 웹 사이트에 프로젝트를 위한 공용 폴더를 만드는 것이다. 로컬 저장소와 구분하는 개념으로 원격저장소라고 한다. GitHub에서는 원격 저장소를 레포지토리(Repository)라고 부른다.

01) GitHub(https://github.com)에 접속하고 로그인한 후 [Create repository] 버튼이나 [New repository] 메뉴를 선택한다.

02) 어떤 장소를 만들지 세부 항목을 작성하는 페이지가 나온다. [Repository name]에 'tshirt'라고 입력한다. 이 이름으로 원격 저장소를 만들겠다는 의미이다. [Description]에는 생성할 원격 저장소에 대한 간단한 설명 '티셔츠 쇼핑몰'이라고 입력한다. 나머지 옵션은 별도로 변경하지 않고, [Create repository] 버튼을 클릭한다.

03) 원격 저장소는 아래와 같다. 아래 주소를 통해 원격 저장소에 접속할 수 있다. 다른 개발자와 함께 작업한다면 이 주소를 알려준다.



## • 원격저장소에 커밋 올리기

이제 GitHub에서 만든 [tshirt] 원격저장소 주소를 내컴퓨터의 [c:/]-[home]-[tshirt] 로컬 저장소에 알려주고 로컬저장소에서 만들었던 커밋들을 원격 저장소에 올려 본다.

01) [c:/]-[home]-[tshirt] 폴더의 Git Bash로 들어온다. 'remote add origin' 명령어로 로컬저장소에 원격 저장소 주소를 알려준다.

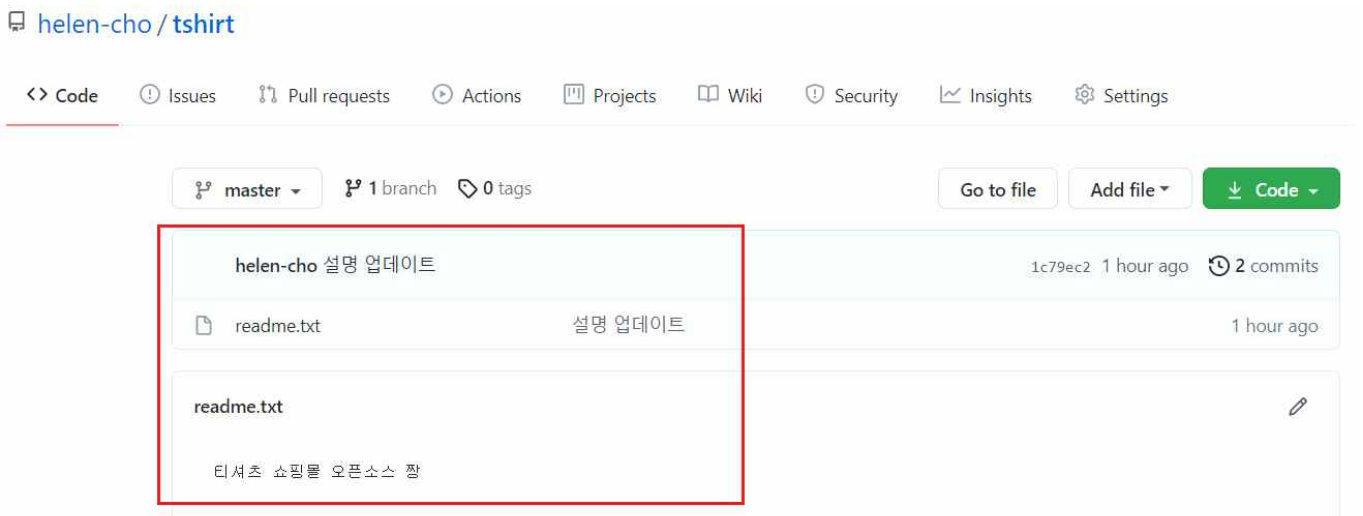
- 원격저장소 확인 : git remote -v
- 원격저장소 삭제 : git remote remove 저장소명
- 원격저장소 이름변경: git rename old저장소명 new저장소명

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git remote add origin https://github.com/helen-cho/tshirt.git
```

02) 로컬저장소에 있는 커밋들을 push 명령어로 원격저장소에 올린다. GitHub 로그인 창에 계정정보를 입력하고 로그인한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git push origin master
git: 'credential-manager-core' is not a git command. See 'git --help'.
git: 'credential-manager-core' is not a git command. See 'git --help'.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 547 bytes | 547.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/helen-cho/tshirt.git
 * [new branch]      master -> master
```

03) GitHub의 원격저장소에서 커밋한 readme.txt 파일이 올라왔는지 확인해 본다.





## • Git Bash에서 로그인한 계정 변경하는 법

1) [제어판] - [사용자 계정] - [자격 증명 관리]에 들어간다.

윈도우는 여기서 사용자의 인증 정보를 보관한다. git도 여기서 관리한다. 기존의 git 계정 정보를 삭제한다.



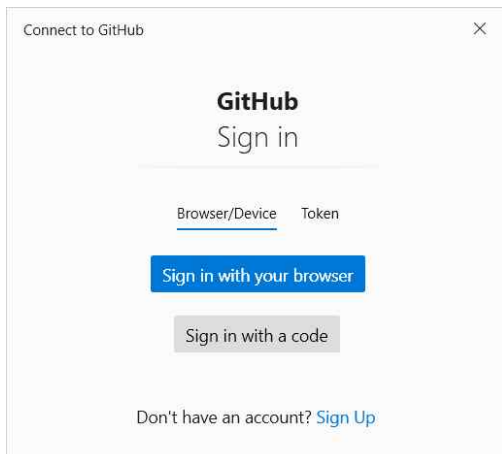
2) [Windows 자격 증명]에서 기존 계정과 관련된 것을 지워준다.

### 자격 증명 관리

웹 사이트, 연결된 응용 프로그램 및 네트워크에 대해 저장된 로그인 정보를 보고 삭제합니다.



3) push 명령어를 입력하고 로그인한다.



## • 원격저장소의 커밋을 로컬저장소에 내려 받기

원격저장소의 코드와 버전 전체를 내 컴퓨터로 내려 받는 것을 클론(clone)이라고 한다. 클론을 하면 최신 버전뿐만 아니라 이전 버전들과 원격저장소 주소 등이 내 컴퓨터의 로컬저장소에 저장된다.

01) [c:/]-[office]-[tshirt] 폴더를 만들어 준다. home에서 올렸던 커밋들을 이곳으로 내려 받는다.

02) [c:/]-[office]-[tshirt] 폴더에서 오른쪽 마우스를 누르고 [Git Bash Here]을 클릭한다.

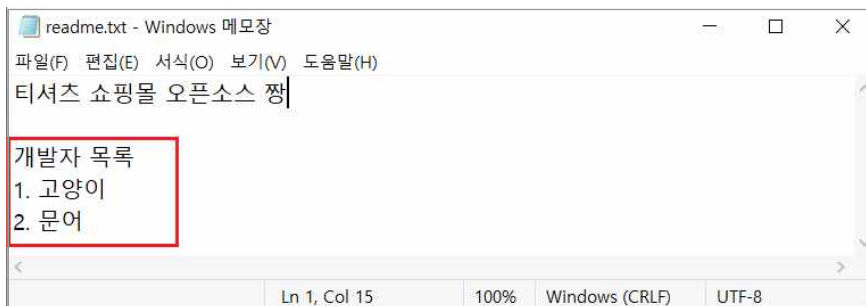
03) clone 명령어 뒤에 원격저장소 주소를 입력하면 어느 원격저장소든 내 컴퓨터의 저장소에 내려 받을 수 있다. git clone 명령어 뒤에 원격 저장소 주소를 적고 한 칸 띄고 마침표를 찍어준다. 마침표는 현재 폴더에 받겠다는 뜻이다. 점을 찍지 않으면 [tshirt] 폴더가 생긴다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/office/tshirt
$ git clone https://github.com/helen-cho/tshirt.git .
Cloning into '.':
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 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 527 bytes | 16.00 KiB/s, done.
```

04) [c:/]-[office]-[tshirt] 폴더에 readme.txt 파일과 [.git] 폴더가 보이면 성공이다.

이름	수정한 날짜	유형	크기
.git	2021-06-06 오후 3:51	파일 폴더	
readme.txt	2021-06-06 오후 3:51	텍스트 문서	1KB

05) [c:/]-[office]-[tshirt] 폴더에 readme.txt 파일을 아래처럼 수정하고 저장한다.



06) Git bash에서 아래 세 개의 명령어를 실행한다. 세 번째 명령어 실행 후에 100% 메시지가 나오면 성공이다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/office/tshirt (master)
$ git add readme.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/office/tshirt (master)
$ git commit -m "개발자 목록 추가"
[master 96/a61f] 개발자 목록 추가
1 file changed, 5 insertions(+), 1 deletion(-)

jhh90@DESKTOP-ILLPH7M MINGW64 /c/office/tshirt (master)
$ git push origin master
git: 'credential-manager-core' is not a git command. See 'git --help'.
git: 'credential-manager-core' is not a git command. See 'git --help'.
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), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/helen-cho/tshirt.git
1c79ec2..967a61f master -> master
```

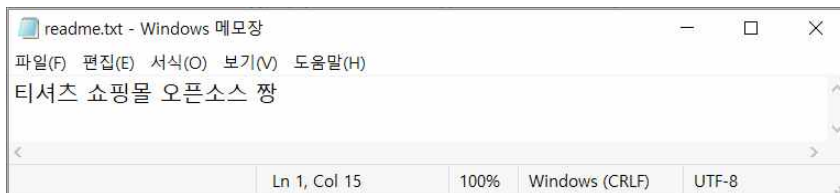
06) GitHub의 원격저장소에 들어가서 새로고침을 해보면 우리가 푸쉬한 '개발자 목록 추가' 커밋이 올라가 있는 것을 확인할 수 있다.



## • 원격저장소의 새로운 커밋을 로컬저장소에 갱신하기

[c:/]-[office]-[tshirt] 폴더에서 새로운 커밋을 만들어서 원격 저장소에 올렸다. 따라서 [c:/]-[office]-[tshirt] 폴더에 있는 readme.txt 파일은 '개발자 목록' 커밋이 반영되어있다. 그러나 [c:/]-[home]-[tshirt] 폴더에 있는 텍스트 파일은 그렇지 않다.

01) [c:/]-[home]-[tshirt] 폴더로 가서 readme.txt 파일을 확인해 본다. 세 번째 커밋이 반영되지 않은 옛날의 두 번째 '설명 업데이트' 커밋의 상태에 머물러 있음을 알 수 있다.



02) [c:/]-[home]-[tshirt] 폴더에서 마우스 오른쪽 버튼을 클릭한 후 [Git Bash Here]를 선택한다.

03) 아래 명령어를 입력한다. pull은 원격저장소에 새로운 커밋이 있다면 그걸 내 로컬 저장소에 받아 오라는 명령어이다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/home/tshirt (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), 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), 329 bytes | 15.00 KiB/s, done.
From https://github.com/helen-cho/tshirt
* branch                master       -> FETCH_HEAD
   1c79ec2..967a61f      master       -> origin/master
Updating 1c79ec2..967a61f
Fast-forward
 readme.txt | 6 +++++-
 1 file changed, 5 insertions(+), 1 deletion(-)
```

04) [c:/]-[home]-[tshirt] 폴더에서 readme.txt 파일을 열어서 확인해 본다. '개발자 목록 추가' 커밋의 상태로 파일이 갱신된 것이다.



## • 깃허브를 이용한 협업방법

이번에는 깃허브 자료를 여러 컴퓨터에 복제하고 함께 사용 및 동기화하여 협업에 사용하는 방법을 학습한다.

### • 다른 컴퓨터에서 원격 저장소 복제하기


01) [c:/]-[cat] 폴더를 생성하고 Git Bash를 실행한 후 git clone 깃허브 주소 명령어를 실행하여 깃을 복제한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat
$ git clone https://github.com/helen-cho/tshirt.git
Cloning into 'tshirt' ...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 844 bytes | 14.00 KiB/s, done.
```

02) [c:/]-[oct] 폴더를 생성하고 Git Bash를 실행한 후 git clone 깃허브 주소 명령어를 실행하여 깃을 복제한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct
$ git clone https://github.com/helen-cho/tshirt.git
Cloning into 'tshirt' ...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 844 bytes | 19.00 KiB/s, done.
```

03) [cat]과 [oct] 폴더안에 [tshirt]라는 폴더가 생겼다. 이 폴더 안에 파일, 깃로그, 원격저장소도 복제 된 것을 확인한다.



```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git log --oneline
967a61f (HEAD -> master, origin/master, origin/HEAD) 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git remote -v
origin https://github.com/helen-cho/tshirt.git (fetch)
origin https://github.com/helen-cho/tshirt.git (push)
```

### • cat이 파일 업로드 기능을 추가하고 oct가 내려 받기 (동기화)

01) [c:/]-[cat]-[tshirt] 폴더에 upload.txt 파일을 작성하고 저장한다.





02) git add upload.txt 명령으로 스테이징 하고 commit 한 후 push한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git add upload.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git commit -m "upload"
[master bdc120c] upload
1 file changed, 1 insertion(+)
create mode 100644 upload.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git push
git: 'credential-manager-core' is not a git command. See 'git --help'.
git: 'credential-manager-core' is not a git command. See 'git --help'.
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), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/helen-cho/tshirt.git
967a61f..bdc120c master -> master
```

03) 깃허브 홈페이지를 새로고침 해서 새로운 커밋이 올라온 것을 확인한다.

master	1 branch	0 tags	Go to file	Add file	Code
helen-cho upload					
bdc120c 3 minutes ago			4 commits		
readme.txt	개발자 목록 추가	1 hour ago			
upload.txt	upload	3 minutes ago			

04) [c:]-[oct]-[tshirt] 폴더에서 내려 받기(pull)를 통해 동기화를 한다.

내 PC > 로컬 디스크 (C:) > oct > tshirt >				
이름	수정한 날짜	유형	크기	
.git	2021-06-06 오후 5:04	파일 폴더		
readme.txt	2021-06-06 오후 5:04	텍스트 문서	1KB	

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git log --oneline
967a61f (HEAD -> master, origin/master, origin/HEAD) 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가
```

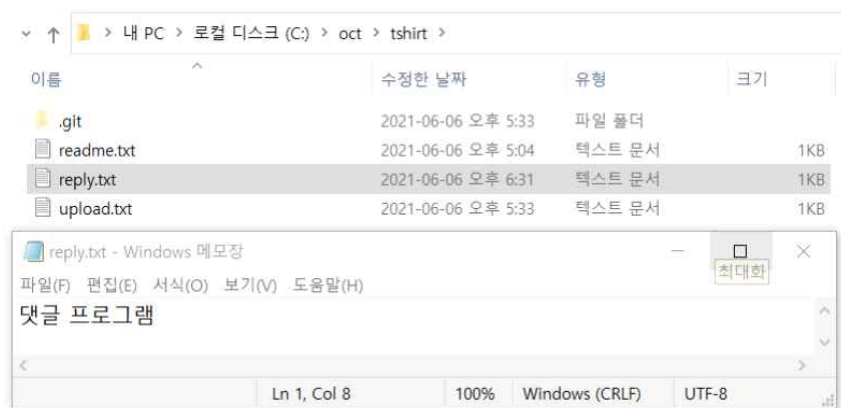
```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 278 bytes | 13.00 KiB/s, done.
From https://github.com/helen-cho/tshirt
967a61f..bdc120c master -> origin/master
Updating 967a61f..bdc120c
Fast-forward
 upload.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 upload.txt
```



## • 원격저장소 변경사항 확인 후 병합하기 (get fetch)

만약 다수가 팀 작업을 할 때 타인이 수정한 원격저장소 최종 커밋을 내려 받기를 하면 나의 PC 폴더가 그대로 동기화된다. 어떤 변화가 있는지 확인 후 동기화 할 경우 fetch명령을 사용한다. fetch는 커밋 정보를 FETCH\_HEAD라는 브랜치로 가져오지만 Merge하지 않는다.

01) [c:]-[oct]-[tshirt] 폴더에 reply.txt 파일을 작성하고 저장한다.



02) git add reply.txt 명령으로 스테이징 하고 commit 한 후 push한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git add reply.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git commit -m "reply"
[master 034dfcb] reply
1 file changed, 1 insertion(+)
create mode 100644 reply.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git push
git: 'credential-manager-core' is not a git command. See 'git --help'.
git: 'credential-manager-core' is not a git command. See 'git --help'.
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), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/helen-cho/tshirt.git
bdc120c..034dfcb master -> master
```

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git log --oneline
034dfcb (HEAD -> master, origin/master, origin/HEAD) reply
bdc120c upload
967a61f 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가
```

03) [c:]-[cat]-[tshirt] 폴더에서 git fetch 명령을 입력한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (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), 298 bytes | 17.00 KiB/s, done.
From https://github.com/helen-cho/tshirt
   bdc120c..034dfcb  master    -> origin/master
```

04) git log --oneline --branches --graph 명령어로 확인 해보면 추가된 커밋과 브랜치는 보이지 않고 폴더에 파일도 보이지 않는다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git log --oneline --branches --graph
* bdc120c (HEAD -> master) upload
* 967a61f 개발자 목록 추가
* 1c79ec2 설명 업데이트
* 014f721 사이트 설명 추가
```

File Explorer view of the directory: > 내 PC > 로컬 디스크 (C:) > cat > tshirt >

이름	수정한 날짜	유형	크기
.git	2021-06-06 오후 6:42	파일 폴더	
readme.txt	2021-06-06 오후 4:58	텍스트 문서	1KB
upload.txt	2021-06-06 오후 5:15	텍스트 문서	1KB

05) git status 명령어로 상태를 확인하면 현재 브랜치가 origin/master에 비해 1개의 커밋이 뒤쳐져 있다고 나온다. 그리고 git pull 명령으로 업데이트 하라고 알려준다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ 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
```

06) git checkout FETCH\_HEAD 명령으로 FETCH\_HEAD 브랜치로 이동한다. git log --oneline 명령으로 cat 커밋이 동기화된 상태를 확인한다. [c:]-[cat]-[tshirt] 폴더를 보면 reply.txt 파일이 존재한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git checkout FETCH_HEAD
Note: switching to 'FETCH_HEAD'.
```

```
Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 034dfcb reply
```

File Explorer view of the directory: > 내 PC > 로컬 디스크 (C:) > cat > tshirt

이름	수정한 날짜	유형	크기
.git	2021-06-06 오후 6:58	파일 폴더	
readme.txt	2021-06-06 오후 4:58	텍스트 문서	1KB
reply.txt	2021-06-06 오후 6:58	텍스트 문서	1KB
upload.txt	2021-06-06 오후 5:15	텍스트 문서	1KB

reply.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

댓글 프로그램

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8



07) fetch한 내용의 확인이 끝나면 브랜치를 합쳐야 한다. 첫 번째 방법은 위에서 이미 설명한 git pull로 내려 받는다. 두 번째 방법은 master 브랜치에서 merge로 브랜치를 합치는 방법이다. git checkout master 명령으로 마스터 브랜치로 이동 후 git merge FETCH\_HEAD 명령으로 브랜치를 합친다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt ((034dfcb...))
$ git checkout master
Previous HEAD position was 034dfcb reply
Switched to branch 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git merge FETCH_HEAD
Updating bdc120c..034dfcb
Fast-forward
 reply.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 reply.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git log --oneline
034dfcb (HEAD -> master, origin/master, origin/HEAD) reply
bdc120c upload
967a61f 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가
```

## • 내PC(지역저장소) 브랜치를 깃허브(원격저장소)에 push하고 깃허브에서 병합(merge)하기

01) [c:]-[cat]-[tshirt] 폴더에서 websocket 브랜치를 만들고 원격저장소로 push한다. (항상 작업 전에 git pull로 동기화를 해준다.)

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git pull
Already up to date.
```

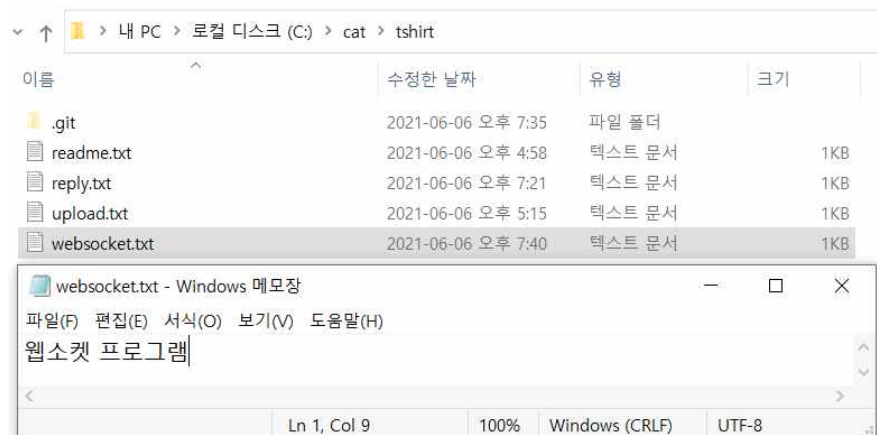
02) git checkout -b websocket 명령어로 websocket 브랜치를 만들고 동시에 checkout으로 브랜치를 이동한다.

• -b 옵션은 뒤에 나오는 브랜치를 만들어서 이동하는 옵션이다.)

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (master)
$ git checkout -b websocket
Switched to a new branch 'websocket'

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (websocket)
$ git log --oneline
034dfcb (HEAD -> websocket, origin/master, origin/HEAD, master) reply
bdc120c upload
967a61f 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가
```

03) [c:]-[cat]-[tshirt] 폴더에 websocket.txt 파일을 생성하고 아래와 같이 내용을 입력한 후 저장한다.





04) git add websocket.txt 명령으로 스테이징, git commit -m "websocket" 명령으로 커밋 한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (websocket)
$ git add websocket.txt

jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (websocket)
$ git commit -m "websocket"
[websocket 3b0f7b8] websocket
1 file changed, 1 insertion(+)
create mode 100644 websocket.txt
```

05) 지역저장소 websocket 브랜치에서 원격저장소 websocket 브랜치로 push 한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (websocket)
$ git push origin websocket
git: 'credential-manager-core' is not a git command. See 'git --help'.
git: 'credential-manager-core' is not a git command. See 'git --help'.
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
```

```
remote:
To https://github.com/helen-cho/tshirt.git
* [new branch]      websocket -> websocket
```

• git log --oneline으로 push 결과를 확인한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/cat/tshirt (websocket)
$ git log --oneline
3b0f7b8 (HEAD -> websocket, origin/websocket) websocket
034dfcb (origin/master, origin/HEAD, master) reply
bdc120c upload
967a61f 개발자 목록 추가
1c79ec2 설명 업데이트
014f721 사이트 설명 추가
```

06) 깃허브 홈페이지에서 새로고침을 해보면 브랜치가 2개로 확인된다. 하지만 마스터 브랜치에서는 websocket.txt 파일이 보이지 않는다.

 master ▾ 2 branches 0 tags

Go to file

Add file ▾

 Code ▾

helen-cho reply

034dfcb 1 hour ago 5 commits

 readme.txt

개발자 목록 추가

4 hours ago

 reply.txt

reply

1 hour ago

 upload.txt

upload

3 hours ago

07) master를 클릭하고 websocket 브랜치로 스위치를 하면 websocket.txt 파일이 나오는 것을 확인할 수 있다.

 websocket had recent pushes 12 minutes ago 

Compare & pull request

websocket ▾ 2 branches 0 tags

Go to file

Add file ▾

 Code ▾

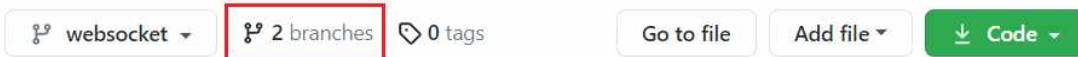
This branch is 1 commit ahead of master.

 Pull request  Compare

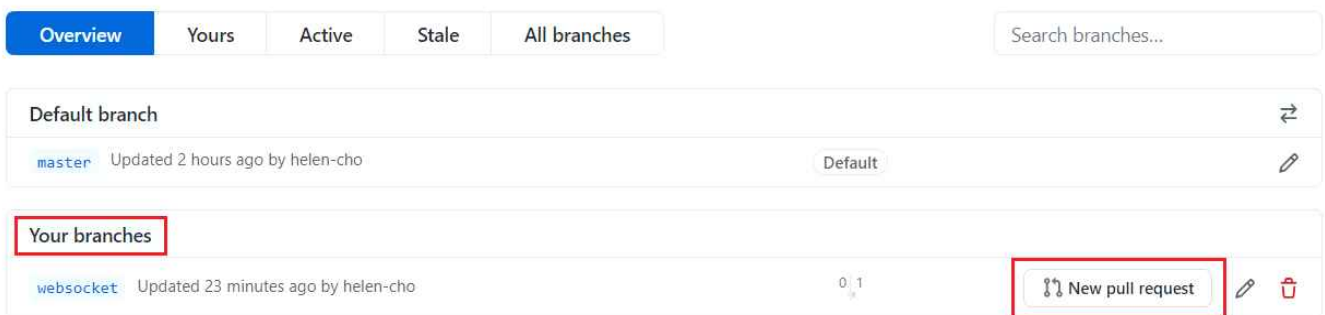
- websocket 브랜치에 websocket.txt 파일이 나오는 것을 확인할 수 있다.

helen-cho websocket		3b0f7b8 16 minutes ago	🕒 6 commits
📄 readme.txt	개발자 목록 추가		4 hours ago
📄 reply.txt	reply		1 hour ago
📄 upload.txt	upload		3 hours ago
📄 websocket.txt	websocket		16 minutes ago

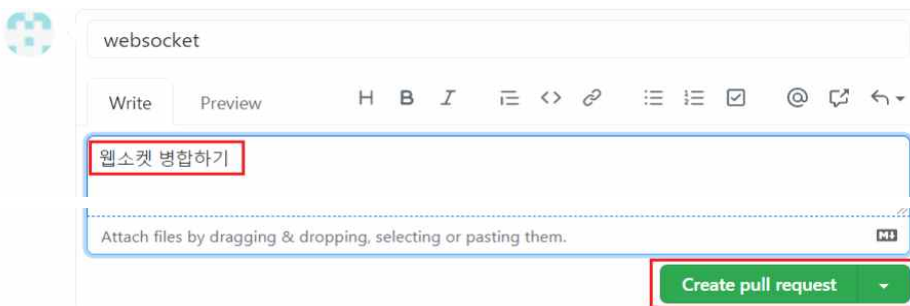
08) 이제 깃허브에서 pull request를 통해 병합을 진행한다. 2 branches를 클릭한다.



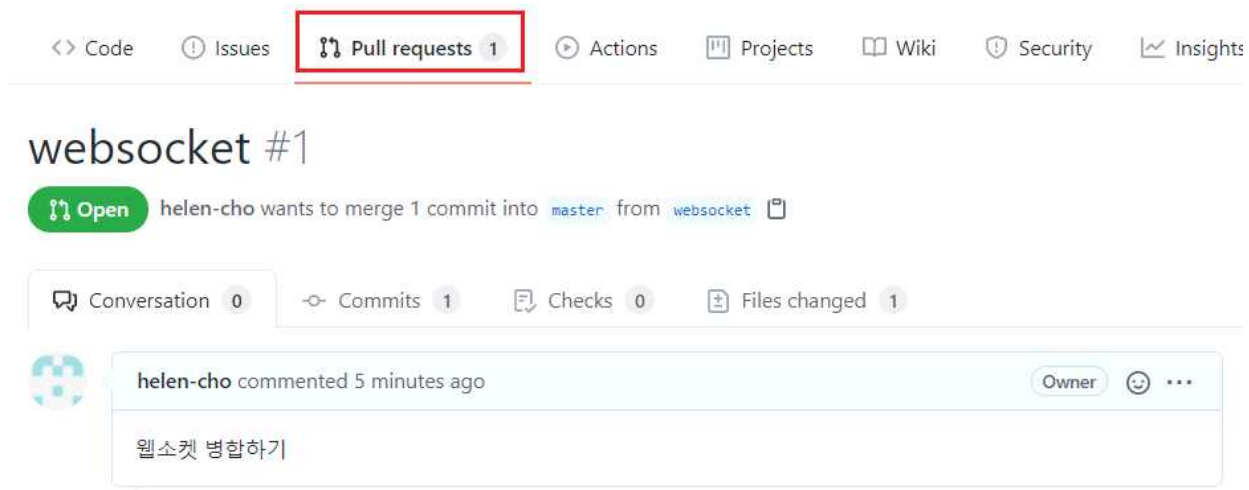
- Your branches의 New pull request를 클릭한다.



- 남길 코멘트를 적고 Create pull request를 클릭한다.



- Pull requests 내용을 확인 한다.



- 내용에 문제가 없다면 Merge pull request를 클릭한다.

- 그리고 Pull requests 내용을 확인 한다.

- <> Code를 클릭하면 master 브랜치에도 websocket.txt 파일이 보인다.

- websocket에서 Merge pull request 됐다는 내용 보인다.

- CLI로 브랜치 관리하기

일반적으로 새로운 기능을 개발하기 위해서 브랜치를 생성하고 개발한 다음 개발이 완료되면 이것을 [master] 브랜치로 병합한다.

1) 현재 생성된 브랜치를 확인한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git branch
* master
  mybranch
  transaction
```

2) 새로운 'upload' 브랜치를 생성 한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git branch upload

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git branch
* master
  mybranch
  transaction
  upload
```

3) 새로운 'upload' 브랜치로 변경한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git checkout upload
Switched to branch 'upload'

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (upload)
$ git branch
  master
  mybranch
  transaction
* upload
```

4) 생성된 'upload' 브랜치를 삭제한다.

```
jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (upload)
$ git branch
  master
  mybranch
  transaction
* upload

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (upload)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git branch -d upload
Deleted branch upload (was 6caa010).

jhh90@DESKTOP-ILLPH7M MINGW64 /c/oct/tshirt (master)
$ git branch
* master
  mybranch
  transaction
```



## • 설정 파일(.gitignore)

.gitignore 파일을 프로젝트 최상위 루트에 생성한 후 그 안에 버전관리에 포함되지 않을 파일을 패턴에 맞게 작성해 준다.

- 표준 Global 패턴을 사용한다.
- 아무것도 없는 라인이나, "#"으로 시작하는 라인은 무시한다.
- 슬래시(/)로 시작하면 하위 디렉터리에 적용되지(Recursivity) 않는다.
- 디렉터리는 슬래시(/)를 끝에 사용하는 것으로 표현한다.
- 느낌표(!)로 시작하는 패턴의 파일은 무시하지 않는다.

### • 설정 예시

```
# 확장자가 .a인 파일 무시
*.a

# 위 라인에서 확장자가 .a인 파일은 무시하게 했지만 lib.a는 무시하지 않음
!lib.a

# 현재 디렉토리에 있는 TODO파일은 무시하고 subdir/TODO처럼 하위디렉토리에 있는 파일은 무시하지 않음
/TODO

# build/ 디렉토리에 있는 모든 파일 무시
build/

# doc/notes.txt 파일은 무시하고 doc/server/arch.txt 파일은 무시하지 않음
doc/*.txt

# doc 디렉토리 아래 모든 .pdf 파일 무시
doc/**/*.pdf
```

### .gitignore에 어떤 조건을 설정해야할지 모를 때 유용한 사이트

.gitignore 내부 설정을 자동으로 생성해 주는 사이트(<https://www.toptal.com/developers/gitignore>)가 존재한다. ex) 프로젝트에서 사용하는 언어, IDE 등을 입력하면 자동으로 .gitignore 파일을 만들어준다.

 | [gitignore.io](https://gitignore.io) 

  
자신의 프로젝트에 꼭 맞는 .gitignore 파일을 만드세요

운영체제, 개발 환경(IDE), 프로그래밍 언어 검색

생성

[소스 코드](#) | [커맨드라인 문서](#)