

GIT 원격 저장소 연동

Git을 제대로 사용하기 위해서는 원격 저장소가 필요하다.

Remote repository(원격 저장소)를 사용하면

- 프로젝트 작업물을 안전 하게 **백업**할 수 있으며,
- 많은 사람들과 **협업**이 용이하다는 장점이 있다.

깃허브(GitHub) 란?

깃을 사용해서 **클라우드를 쓰듯이 소스들을 업로드해서 저장**하고, **다른 개발자들과 공유, 협업** 할 수 있는 깃 저장소들 중에 대표적인 것으로 Bitbucket, Gitlab, Github가 있는데, 그 중 깃허브가 가장 유명하다.

깃과 깃허브에 익숙해진다면,

내 소스들을 안전하게 깃허브에 저장하고, 다른 팀원들과 수월하게 협업을 할 뿐 아니라, 깃허브라는 방대한 코드의 숲에서 전세계 뛰어난 개발자들과 함께하는 경험을 할 수 있을 것이다.

저장소 생성하기 (Create Repository)

1. Remote Repository 생성

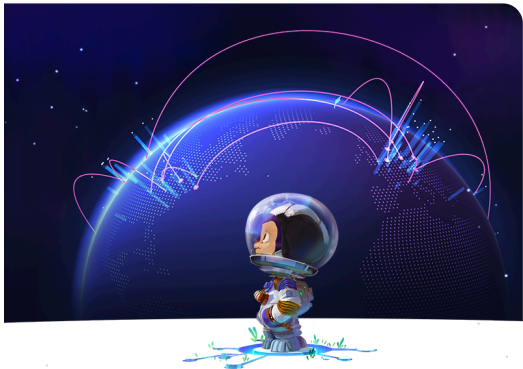
- 깃헙 사이트 접속 및 로그인

<https://github.com/>

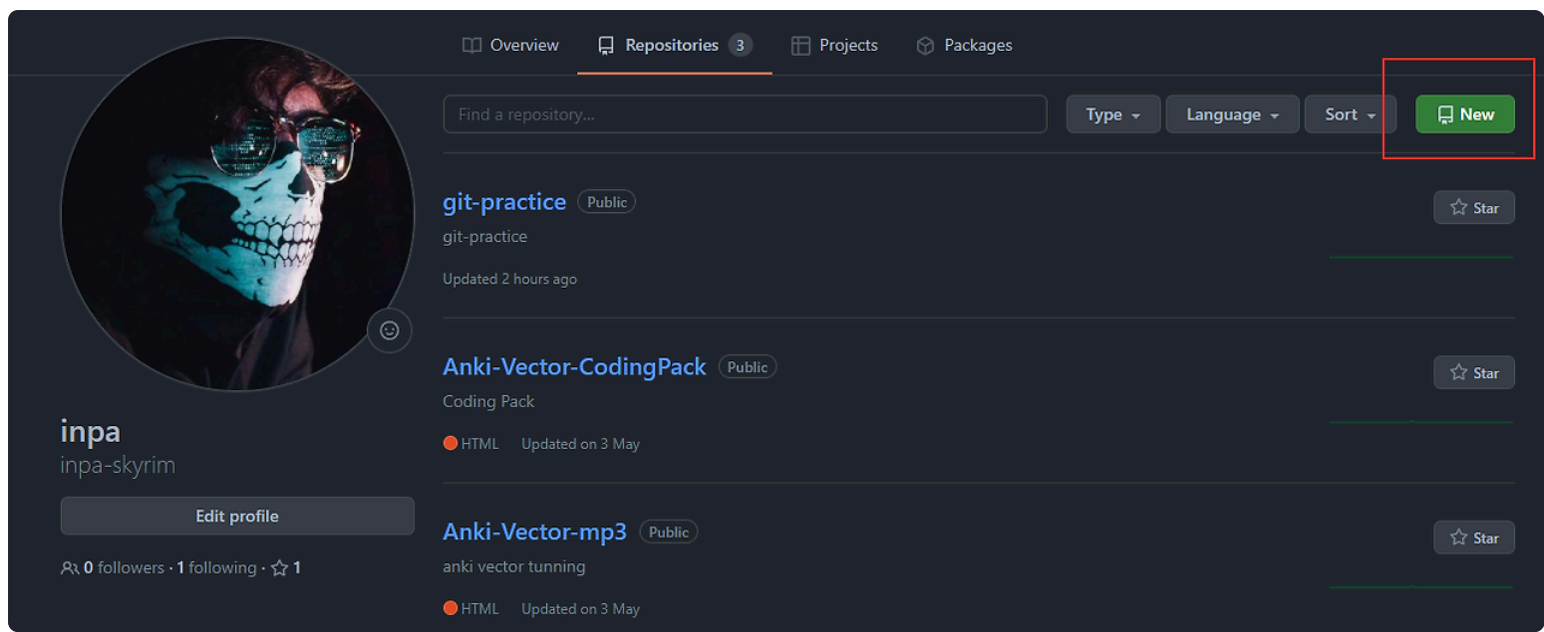
GitHub: Where the world builds software

github.com

GitHub is where over 73 million developers shape the future of software, together. Contribute to the open source community, manage your Git ...



- **New(New Repository)** 클릭



- Create a new repository 내용 작성

Owner *

inpa-skyrim

Repository name *

name

Great repository names are short and memorable. Need inspiration? How about **fantastic-fiesta**?

Description (optional)

참고로 한글은 안됨

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

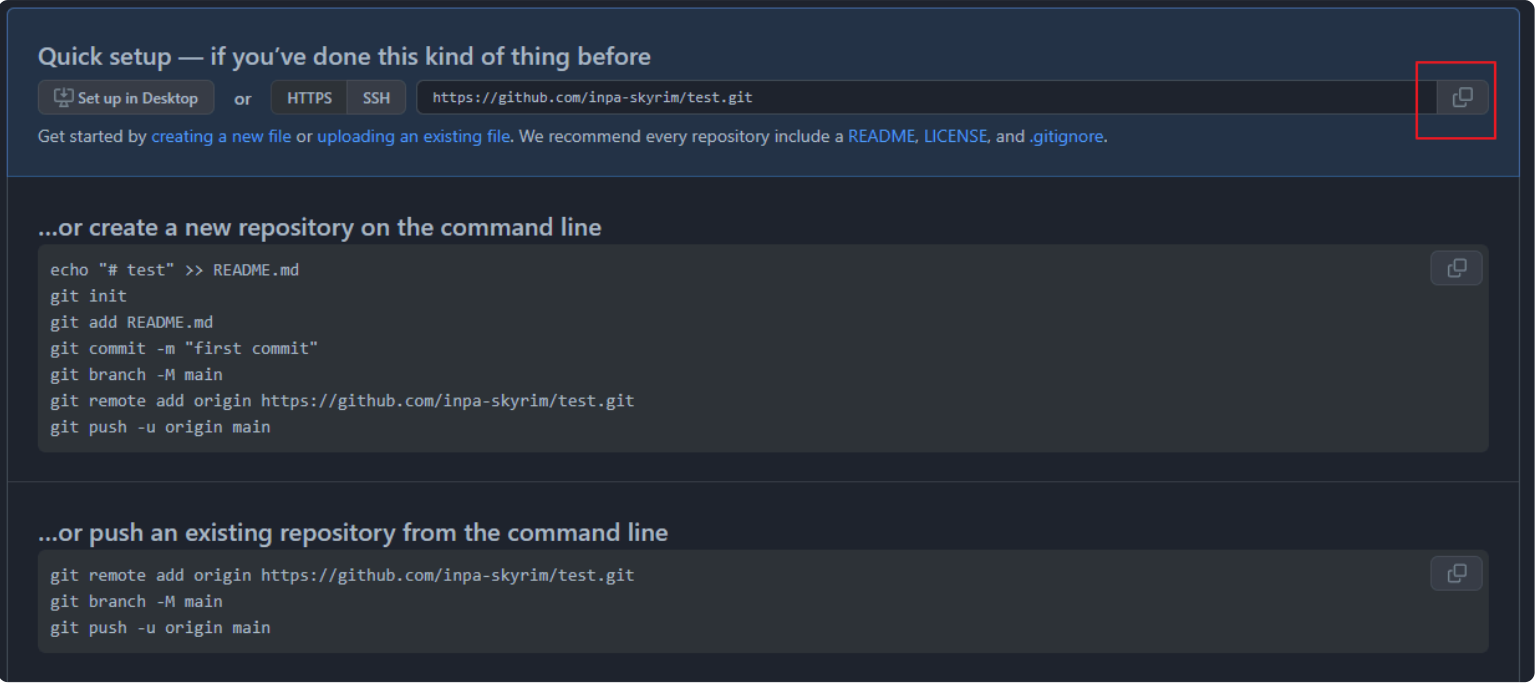
Tip

- 1) Repository name : 앞으로 사용하게될 저장소 이름, 결국 프로젝트 명이 된다. (깃헙 주소 뒤 프로젝트 명이 따라오는 것을 볼 수 있다.)
- 2) 권한설정
 - 공개(Public) : 모든 사용자가 볼 수 있다.
 - 비공개(Private) : 나와, 지정한 사용자만 볼 수 있다.
- 3) "Create Repository" 클릭

2. git remote (Remote Repository 연결)

- **git remote** 명령으로 현재 프로젝트에 등록된 리모트 저장소를 확인할 수 있다.
- 이 명령은 리모트 저장소의 단축 이름을 지어준다.

- URL은 제일 상단 부에서 확인 가능하고, 클립보드에 복사하도록 제공 하기도 한다.



BASH

```
# git remote add <remote repo 이름> <repo url>
$ git remote add origin https://github.com/inpa-skyrim/git-practice.git
# https://github.com/깃헙계정/리포지터리명.git
# url를 origin이라고 이름을 붙여 추가하겠다는 의미
# url가져오기
$ git remote get-url origin
```

```
MS@DESKTOP-PQJEJIL MINGW64 ~/Desktop/공부/git연습 (master)
$ git remote add origin https://github.com/inpa-skyrim/git-practice.git

MS@DESKTOP-PQJEJIL MINGW64 ~/Desktop/공부/git연습 (master)
$ git remote get-url origin
https://github.com/inpa-skyrim/git-practice.git
```

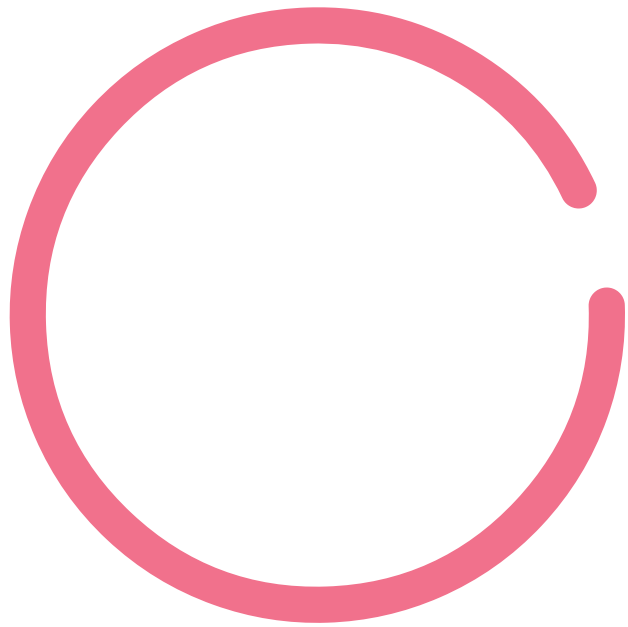
이로써 Remote Repository와 연결은 끝난 것이다.

옵션

BASH

```
# 옵션 종류 보기
$ git remote --help
# 추가한 원격저장소의 목록 확인
$ git remote
$ git remote -v # 상세히
# 특정 원격 저장소의 정보를 확인할 수 있다.
$ git remote show 이름
# 원격저장소 이름 변경
$ git remote rename 기존이름 변경할이름
# 원격저장소를 제거
$ git remote rm 이름
```

Remote 저장소 업로드



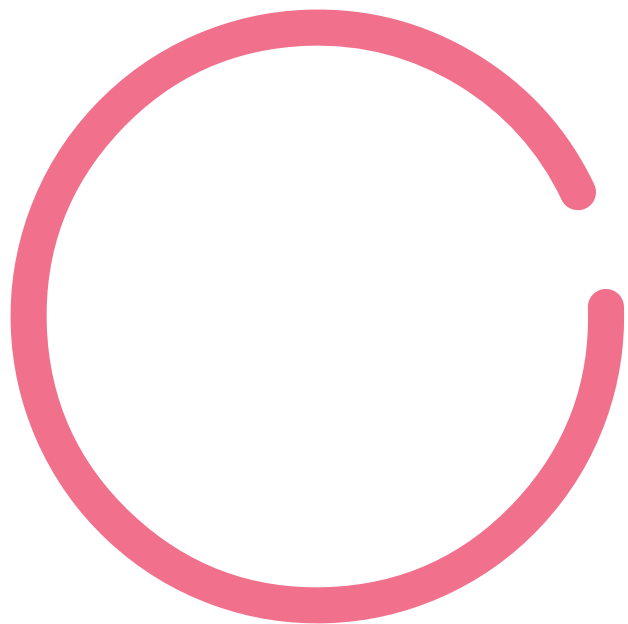
git push

- 현재 branch에서 새로 생성한 **commit**들을 원격 저장소에 업로드
- 로컬 컴퓨터에서 작업하고 커밋을 깃허브에서 온라인으로도 볼 수 있다

내 작업들을 깃헙에 올리기(푸쉬하기)

BASH

```
$ git push -u origin master  
# origin : 원격주소 / master : 브랜치  
# 영어문법 4형식 -> origin에 master를 push하라
```



⚠ 그런데 다른 사람이 깃헙에서 hello3.txt를 수정했다고 치자






깃헙에서 수정하고 커밋을 하면,

로컬에는 해당 커밋이 없는데, 깃헙에서 새로운 커밋이 추가됨을 확인 할 수 있다.

즉, 내 로컬 커밋 로그와, 깃헙 커밋 로그가 달라지게 된다.

이것이 바로 협업 과정에서 생기는 **충돌**이라고 한다.

inpa-skyrim github commit 1	
 .gitignore	fifth commit
 hello1.txt	2 commit
 hello2.txt	3 commit
 hello3.txt	github commit 1

```
$ git shortlog
inpa (4):
  fifth commit
  1 commit
  2 commit
  3 commit
```

만일 로컬(나)이 이 사실을 모르고,

로컬이 hello3.txt를 수정하고 push를 하면 이렇게 에러가 뜬다.

에러 내용을 보면 **push하기전에 pull**하라는 소리가 나온다.

```
.gitignore  hello1.txt  hello3.txt  hello2.txt
git연습 > hello3.txt
You, 3 minutes ago | 1 author (You)
1  누가 여길 바꿔버렸는데 저는 몰라요... | You, seconds ago • Uncommitted changes

문제  출력  터미널  디버그 콘솔

$ git push origin master
To https://github.com/inpa-skyrim/git-practice.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/inpa-skyrim/git-practice.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.

MS@DESKTOP-PQJEJIL MINGW64 ~/Desktop/공부/git연습 (master)
$
```

git pull

- 원격 저장소에서 파일 내려받기
- git pull = git fetch + git merge** : 원격저장소 커밋과 동기화하고 커밋을 머지 시킨다.

- 원격 저장소와 로컬 저장소의 상태를 같게 만들기 위해 원격 저장소의 소스를 가져오는 것이다.
- 즉 다른 사람들의 작업 변경사항을 클라이언트로 내려받기 한다고 보면 된다.

pull을 통해 서버로부터 작업내용을 가져와보자.

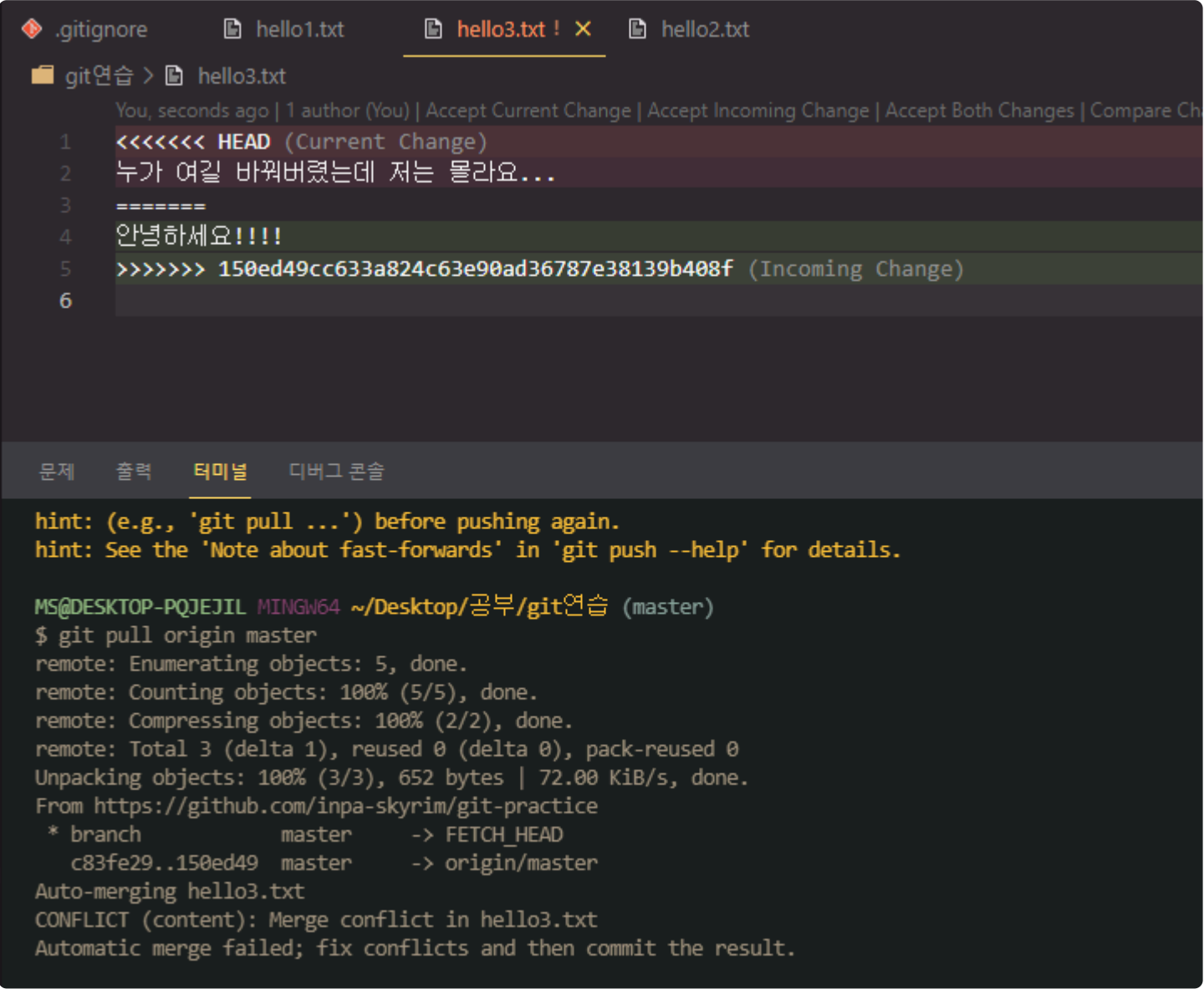
명령어를 실행하면 이렇게 충돌 표시가 나오는걸 확인할 수 있다.

왜냐하면 원격 저장소의 커밋내용과, 로컬 저장소의 커밋내용이 중복되어 겹치기 때문이다.

GIT은 누가 우선순위인지 알지못한다. 따라서 사용자가 고쳐주어야 한다.

BASH

```
# 원격 저장소의 변경사항을 워킹 트리에 반영
# git pull <remote> <branch>
$ git pull origin master
```

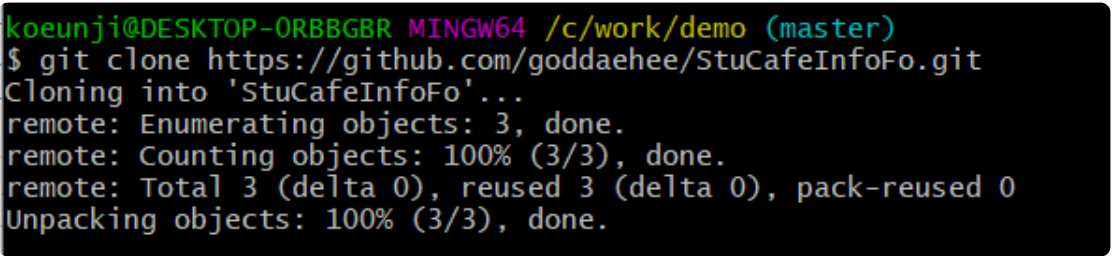


git clone

- 원격 저장소의 저장소를 내 local에서 이용할 수 있게 **그대로 똑같이 복사해 가져온다.**

BASH

```
$ git clone [remote repo 주소]
$ git clone https://github.com/goddaehee/StuCafeInfoFo.git
```



🔔 git pull vs git clone 차이점

git pull 명령은 원격 저장소에 있는 프로젝트 내용을 가져오는 역할을 한다.
그리고 보니 git clone도 원격 저장소에 있는 프로젝트 내용을 가져오는 명령이지 않은가?

git pull과 git clone은 명백한 차이점이 있다.
먼저, git clone명령을 사용하면 **로컬 저장소의 내용이 원격 저장소의 내용과 일치**해진다.
기존에 작업중이었던 사람이 git명령을 사용해서 원격 저장소의 내용을 그대로 가져와버리면 **기존에 작업했던 내용들은 당연히 없어진다.**
즉, git clone은 프로젝트에 처음 투입될 때 사용되어야 하는 명령인 것이다.

반면 git pull명령은 원격 저장소의 내용을 가져와서 현재 브랜치와 **병합(merge)** 해주는데, **그 이외의 작업 파일들은 건들지 않는다.**

git pull명령은 병합과정도 포함되어 있기 때문에, pull을 하기 전에 commit을 하지 않으면 덮어쓰기 에러가 발생할 수 있다.
즉 기존 작업에 대해 commit을 미리 해두고 pull을 수행해주는 습관을 들이자.

git fetch

- 원격 저장소의 branch와 commit들을 로컬 저장소와 동기화. 단 merge는 안함
- 옵션을 생략하면 모든 원격 저장소에서 모든 브랜치를 가지고 옴

BASH

```
$ git fetch [원격 저장소명] [브랜치명]
```

다른 사용자 초대하기

- "Settings"탭 클릭 > Manage access > "invite a collaborator" 클릭

goddaehee / StuCafeInfoFoPrivate

Unwatch1Star0Fork0

<> Code

Issues0

Pull requests0

Actions

Projects0

Wiki

Security

Insights

Settings

Options

Manage access

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Autolink references

Secrets

Actions

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Beta

Learn more or give us feedback

Manage access

You haven't invited any collaborators yet

If you're using GitHub Free, you can add unlimited collaborators on public repositories, and up to three collaborators on private repositories owned by your personal account. [Learn more](#)

Invite a collaborator

- 초대한 사용자 입력, 선택 하기



Invite a collaborator to **StuCafeInfoFo**



gitkozee

Invite collaborator

- 메일을 확인하면 해당 repository invite 메일이 와있을 것이다. "View invitation" 클릭

GitHub



@goddaehee has invited you to collaborate on the **goddaehee/StuCafeInfoFo** repository

You can [accept](#) or [decline](#) this invitation. You can also visit [@goddaehee](#) to learn a bit more about them.

This invitation will expire in 7 days.

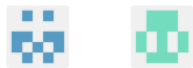
[View invitation](#)

- 초대 수락 클릭 ("Accept invitation" Click)

goddaehee / **StuCafeInfoFo** Private

Watch 1 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights



goddaehee invited you to collaborate

[Accept invitation](#)

[Decline](#)

Tip

※ 초대 받은 사용자 또한 위의 명령어(remote, clone)를 통해 원격 저장소를 연결 해보자.

원격 저장소 연결 및 끊기

먼저 git remote -v 명령어를 사용하여 현재 연결되어 있는 원격 레파지토리를 확인

BASH

```
$ git remote -v
origin  https://github.com/IfUwanna/Tool.git (fetch)
```


origin https://github.com/IfUwanna/Tool.git (push)

기존 리포지토리 remote 제거

해당 원격 저장소의 연결을 제거하기 위해서

git remote remove <name> 옵션을 사용해 주면 연결되어 있는 저장소를 간단하게 끊을 수 있다.

BASH

```
# git remote remove <name>
$ git remote remove origin
```

이후 다시 -v 옵션을 통해 확인해 보면 연결된 원격 저장소가 없는 것을 확인 할 수 있다.

새 리포지토리 remote 추가

다시 원격저장소와 연결하고 싶은 경우엔 다시 git remote add <name> <url> 옵션을 사용하면 된다.

BASH

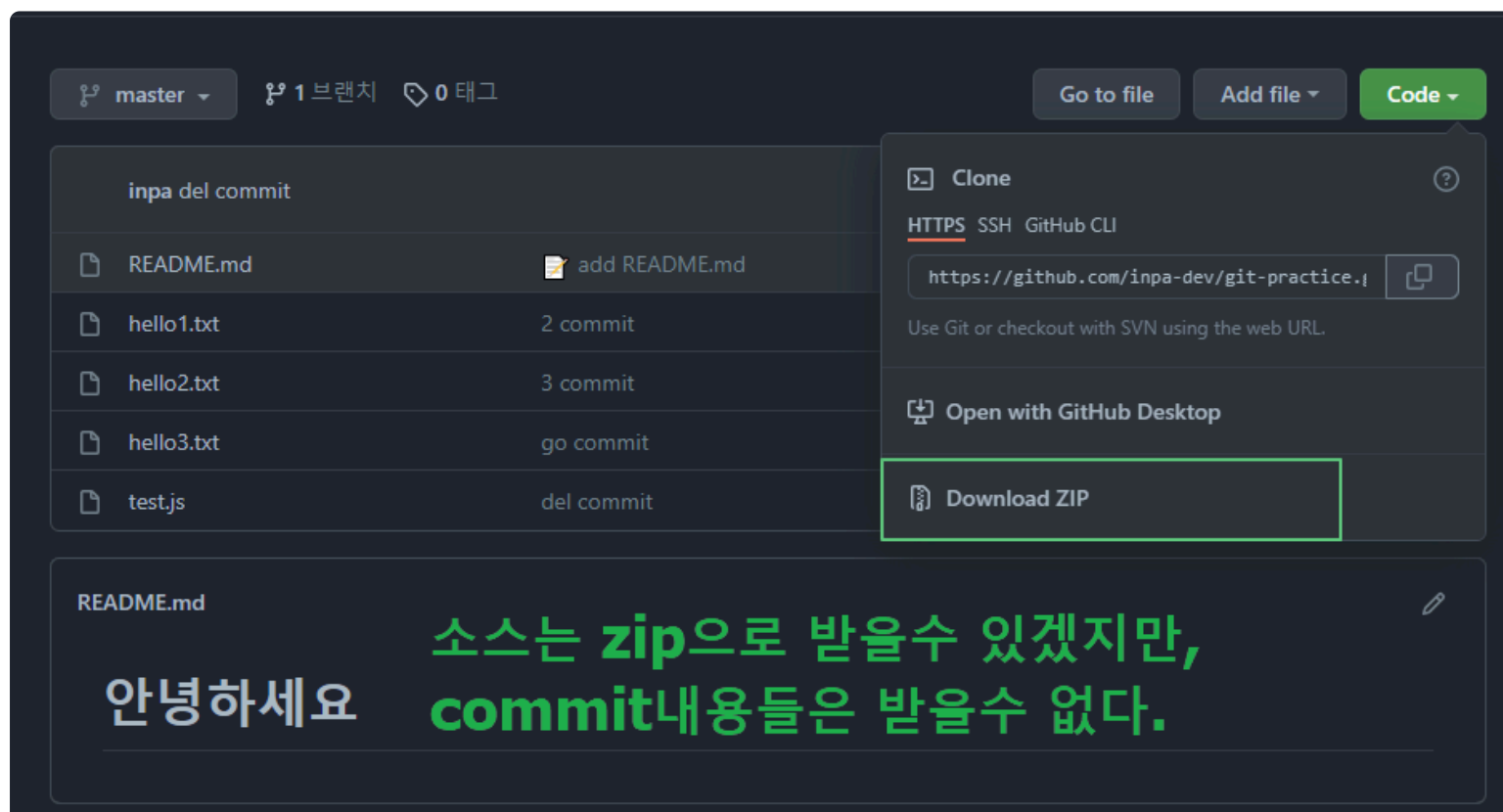
```
# git remote add <name> <url>
$ git remote add origin https://github.com/IfUwanna/Tool.git
```

깃헙에서 소스 및 커밋 내려 받기

예를 들어 회사에 있는 컴퓨터로 작업 후 깃허브에 올린 후 집에 와서 깃허브에서 소스를 내려받아 이어 작업한다고 가정해 보자.

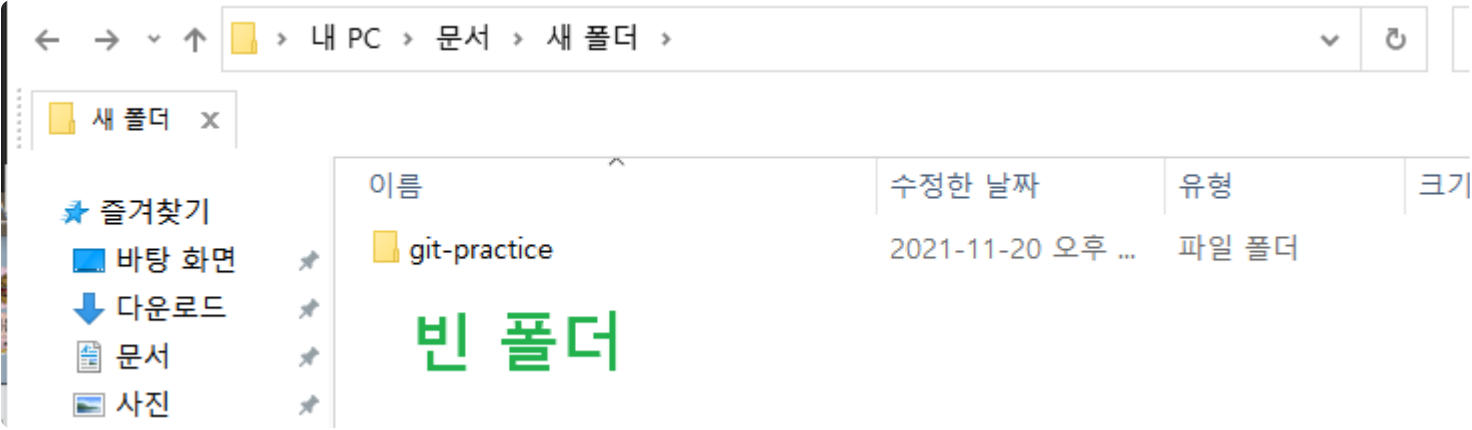
보통 바로 다운받는 방법을 쓸 것이다.

하지만 크게 착각하는 부분이 있는데, 이렇게하면 **git으로 작업할수 없다는 것이다.**



소스 와 commit내용까지 모두 받아주기 위해, git명령어를 통해 받아오자

원하는 경로로 가서 새 폴더를 만들어준다.

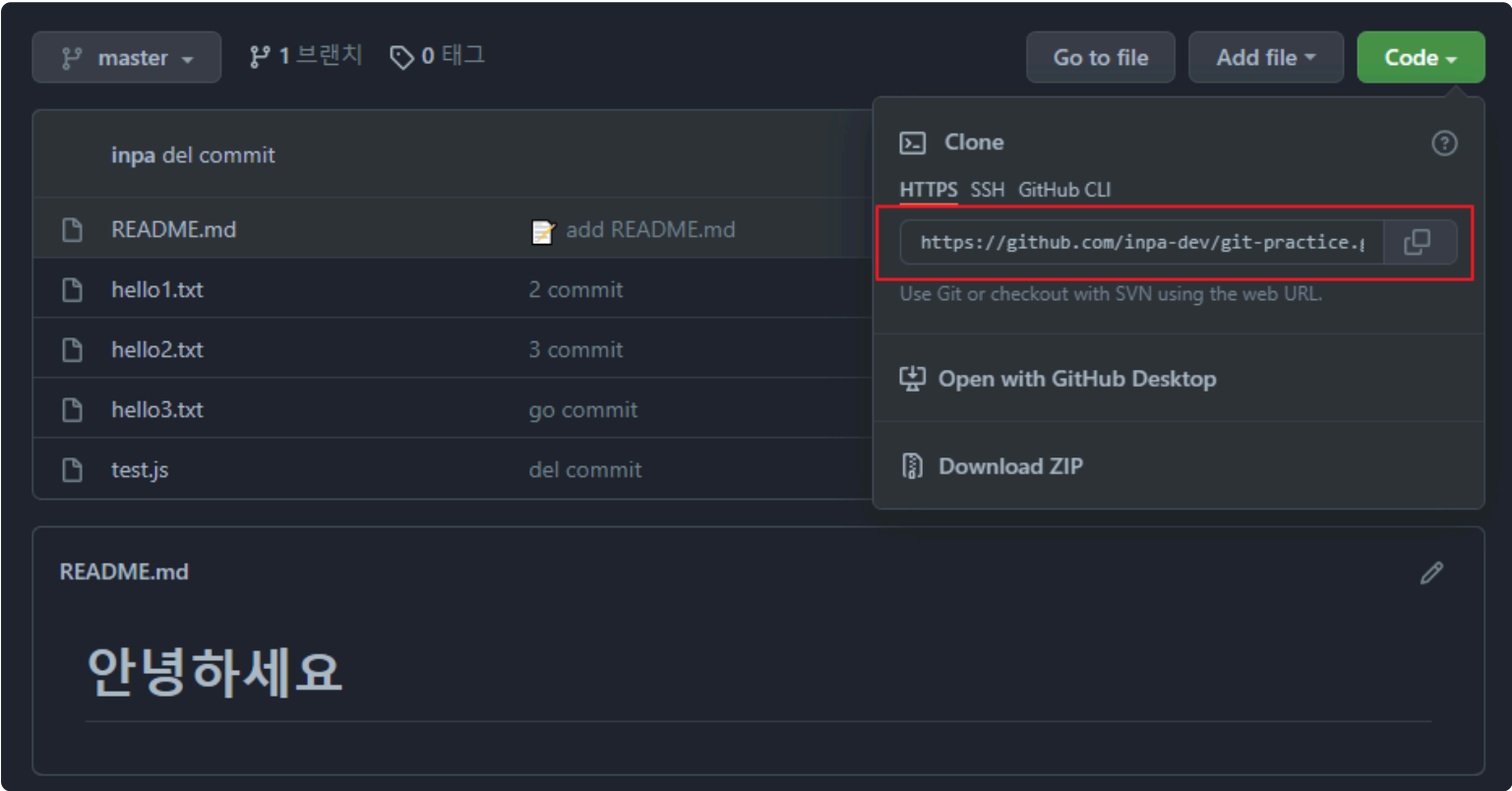


cmd에서 해당 경로로 이동해준다. 그리고 다음 명령어를 입력한다.

git url은 빨간 박스에서 가져오면 된다.

BASH

```
$ git clone https://github.com/유저네임/원격저장소이름.git
```



```
C:\Users\barzz\OneDrive\문서\새 폴더
λ git clone https://github.com/inpa-dev/git-practice.git
Cloning into 'git-practice'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 41 (delta 10), reused 31 (delta 5), pack-reused 0
Receiving objects: 100% (41/41), 4.90 KiB | 1.63 MiB/s, done.
Resolving deltas: 100% (10/10), done.
```

협업 작업시 주의할 사항

팀원과 작업할 때는, 먼저 **fetch**와 **status**를 통해 **pull**할 내용이 있는지 먼저 확인하는 것이 중요하다.

BASH

```
$ git fetch
```

```
$ git status
```

```
Gachon1@DESKTOP-2LGSDH3 MINGW64 /c/Projects/Projects_Python/gitTest (master)
$ git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 322 bytes | 21.00 KiB/s, done.
From https://github.com/.../gitTest
  579bd2e..ae7ee3e  master    -> origin/master

Gachon1@DESKTOP-2LGSDH3 MINGW64 /c/Projects/Projects_Python/gitTest (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

Gachon1@DESKTOP-2LGSDH3 MINGW64 /c/Projects/Projects_Python/gitTest (master)
$
```

이 브랜치가 원격 origin의 마스터에 커밋 하나가 뒤쳐져 있다는 메시지를 볼 수 있다.

이렇게 분업을 하는 경우 pull할 내용이 있을 경우 pull해주지 않으면 로컬에서 작업 후 push를 할 수 없다.

BASH

```
# 원격저장소와 로컬저장소 commit을 일치시키고 merge 시킨다.
$ git pull <원격저장소명> <브랜치명>
```