

GitHub

허브?



git와 github의 관계는?

영상카메라 → 너튜브, 네TV, 카TV

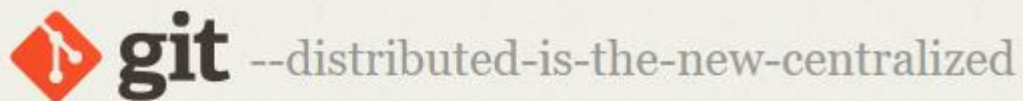
(촬영도구)

(촬영 영상의 공유장소)

git → GitHub

(코드 관리 툴)

(코드 공유 장소)



Git is a [free and open source](#) distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is [easy to learn](#) and has a [tiny footprint with lightning fast performance](#). It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like [cheap local branching](#), convenient [staging areas](#), and [multiple workflows](#).



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases



Community

Get involved! Bug reporting.

Latest source Release

2.30.1

[Release Notes](#) (2021-02-08)

[Download 2.30.1 for Windows](#)



🔍 Search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloading Git



Your download is starting...

You are downloading the latest (**2.30.1**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **21 days ago**, on 2021-02-09.

[Click here to download manually](#), if your download hasn't started.

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

The current source code release is version **2.30.1**. If you want the newer version, you can build it from [the source code](#).



Git-2.30.1-64-bit.exe

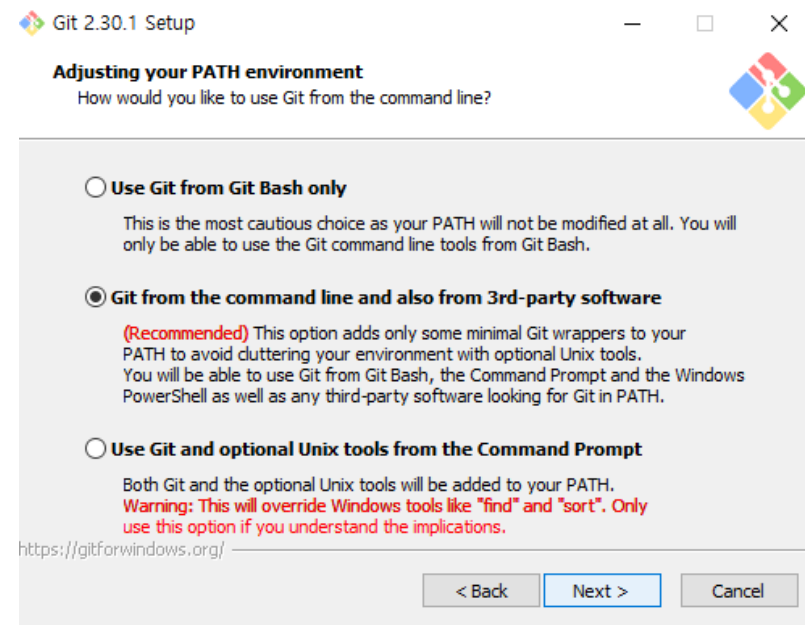
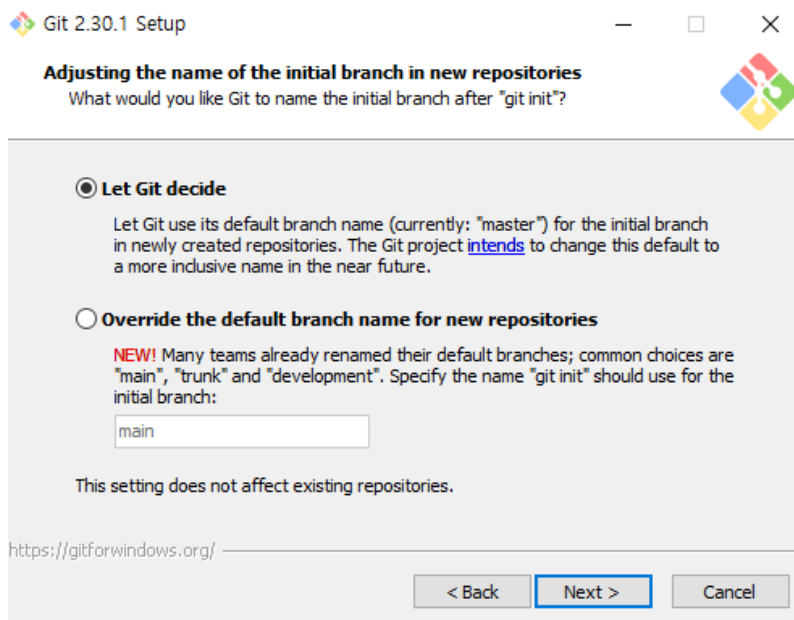
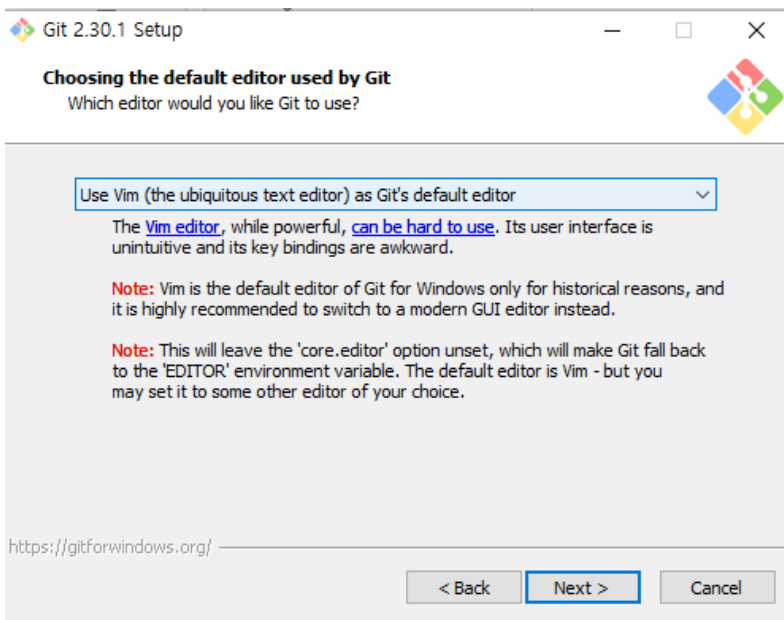
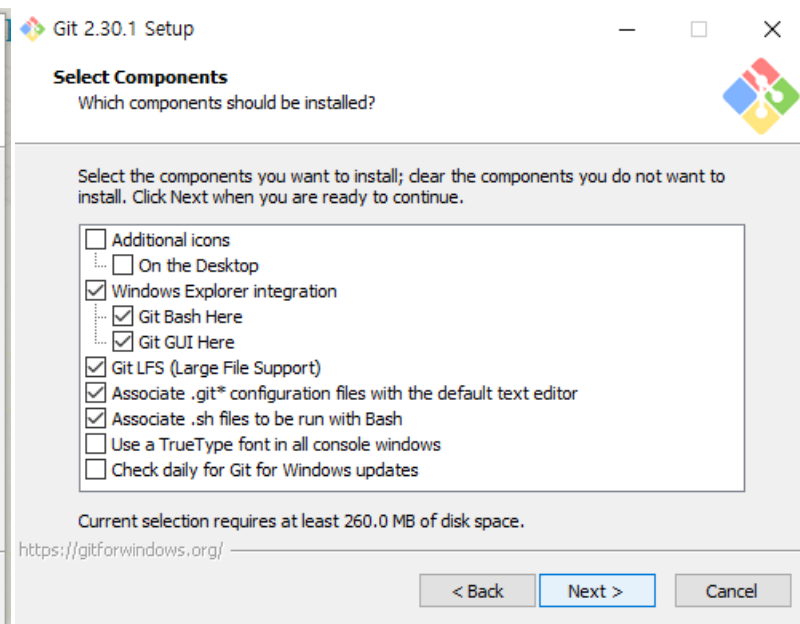
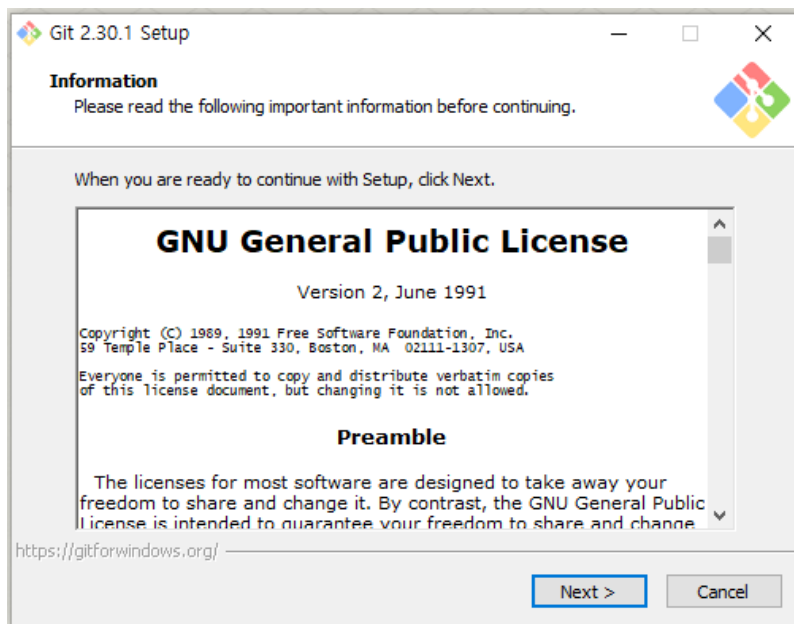
Opening in 5 mins...



설치

그냥 다

Next만 누르면 돼요!



git 깃 = 코드 관리를 위한 도구

코드 합치기 (조각 + 조각 이미 완성 해뒀던 것들)

코드 개선하기 (고치지만, 이전 버전 돌아갈지도 ?)

코드 배포하기

코드 잘 검색하기

코드 변경사항 기록

git 깃 역할

코드 병합 (merge, rebase)

코드 리비전 관리 (reset, commit, branch)

코드 릴리즈 (push)

코드 태깅 (tag)

코드 변경사항 검토 (diff, log)

git 깃 역할

소스 코드 병합 (merge, rebase)

소스 리비전 관리 (reset, commit, branch)

소스 릴리즈 (push)

소스 태깅 (tag)

소스 변경사항 검토 (diff, log)

git

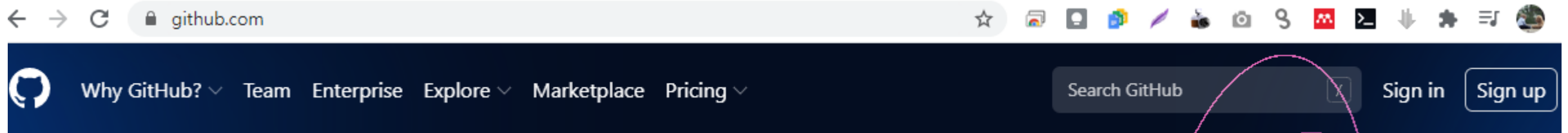
처음 시작하는 것이라면 git의 config 과정을 진행해야합니다.
git config 명령어를 이용하여 계정에 대한 정보를 설정합니다.

```
$ git config --global user.name "shongdr"
```

```
$ git config --global user.email "shongdr@gmail.com"
```

깃 시작 git init 작업 진행(GitHub에서 클론을 받은경우 필요x)
다음장에..

GitHub 가입



- Sign Up: Sign 을 up업로드
- Sign in: Sign 으로 in들여가기

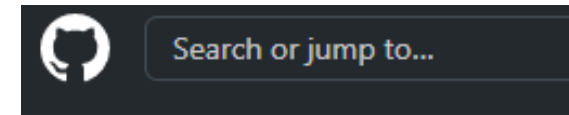
GitHub Repository 생성

1. 깃허브 사이트 로그인

2. New 버튼 or

오른쪽 상단 (2시방향)에 있는 +버튼을 눌러서

New repository 버튼을 눌러줍니다.

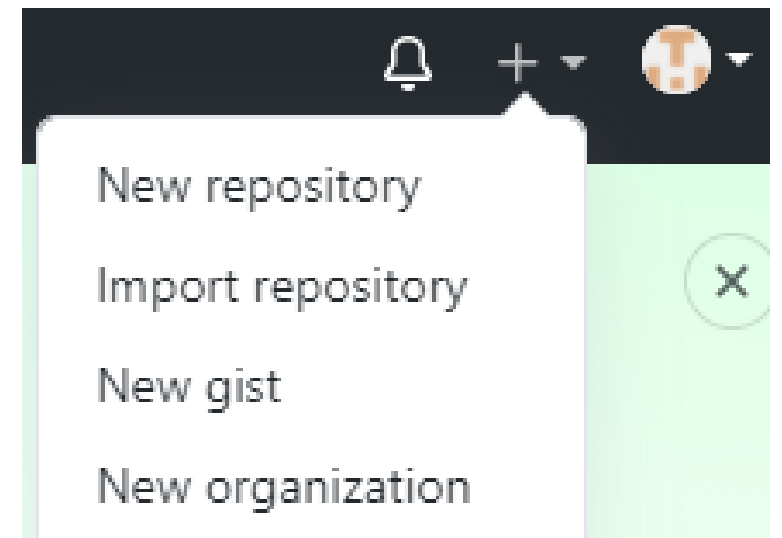


Repositories

New

Find a repository...

shongdr/pyex



GitHub Repo

1. 공유 공간 이름

2. 공개 여부 결정

Create a new repository

A repository contains all project files, including the revision history. Already have a repository? [Import a repository.](#)

Owner *



shongdr ▾

Repository name *

pyex-librosa-animalforest ✓

Great repository names are short, lowercase, and hyphenated. [pyex-librosa-animalforest](#) is available. [Learn more about repository names.](#)

Description (optional)



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

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



Set up in Desktop

or

HTTPS

SSH

<https://github.com/shongdr/pyex-librosa-animalforest.git>

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

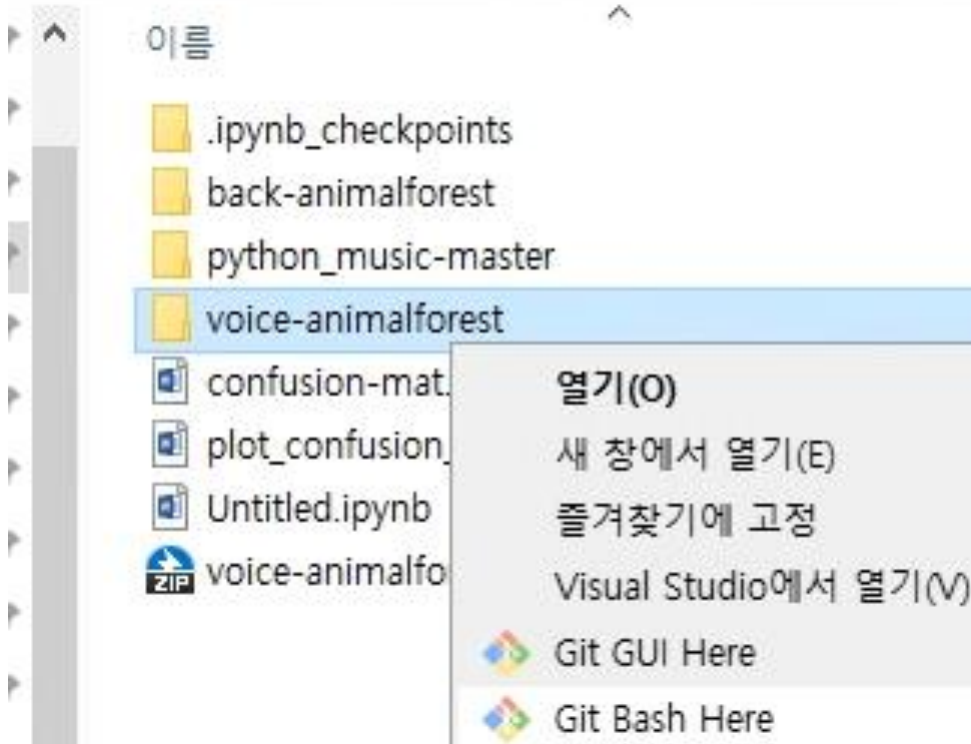
...or create a new repository on the command line

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

git Bash

1. 공유하고 싶은 폴더
2. 우측 클릭
3. Git Bash Here 클릭

Administrator > lect_py > pyex >



MINGW64:/c/Users/Administrator/lect_py/pyex/voice-animalforest

```
Administrator@DESKTOP-61JS5V0 MINGW64 ~/lect_py/pyex/voice-animalforest
$
```


git init → (git status) 상태보기 = 생략 가능

\$ git init

```
MINGW64:/c/Users/Administrator/lect_py/pyex/voice-animalforest
Administrator@DESKTOP-61J55V0 MINGW64 ~/lect_py/pyex/voice-animalforest
$ git init
Initialized empty Git repository in C:/Users/Administrator/lect_py/pyex/voice-animalforest/.git/
```

내컴퓨터 폴더 → 내컴퓨터에 업로드 등록할 '빈공간' 준비 !

\$ git status

'올릴 거라고 등록' 상태 확인

붉은색 = 미등록 x

녹색 = 등록 o

```
MINGW64:/c/Users/Administrator/lect_py/pyex/voice-animalforest
Administrator@DESKTOP-61J55V0 MINGW64 ~/lect_py/pyex/voice-animalforest (master)
$ git status
On branch master
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        README.md
        aniforest-librosa-pitch-lect.ipynb
        aniforest-librosa-pitch.ipynb
        aniforest_librosa.ipynb
        audio-processing-using-librosa-for-beginners.ipynb
        ffmpeg/
        ffmpeg_20201004/
        librosa.ipynb
        librosa2.ipynb
        main-jeon.ipynb
        main.ipynb
        main.py
```

git add → (git status) 상태보기 = 생략 가능

\$ git add .

```
Administrator@DESKTOP-61J55V0 MINGW64 ~/lect_py/pyex/voice-animalforest (master)
$ git add .
```

내컴퓨터 폴더 → 업로드 대상 등록

\$ git status

‘올릴 거라고 등록’ 상태 확인

붉은색 = 미등록 x

녹색 = 등록 o

```
Administrator@DESKTOP-61J55V0 MINGW64 ~/lect_py/pyex/voice-animalforest (master)
$ git status
On branch master

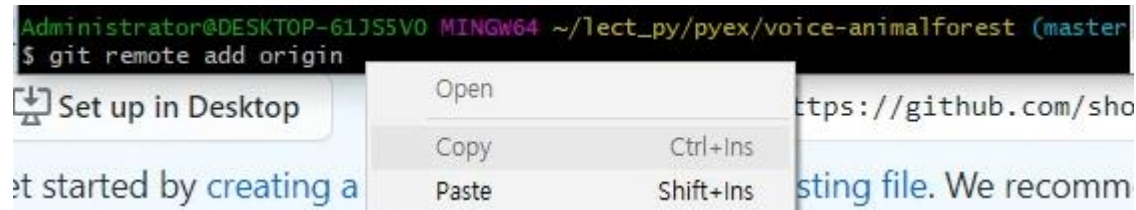
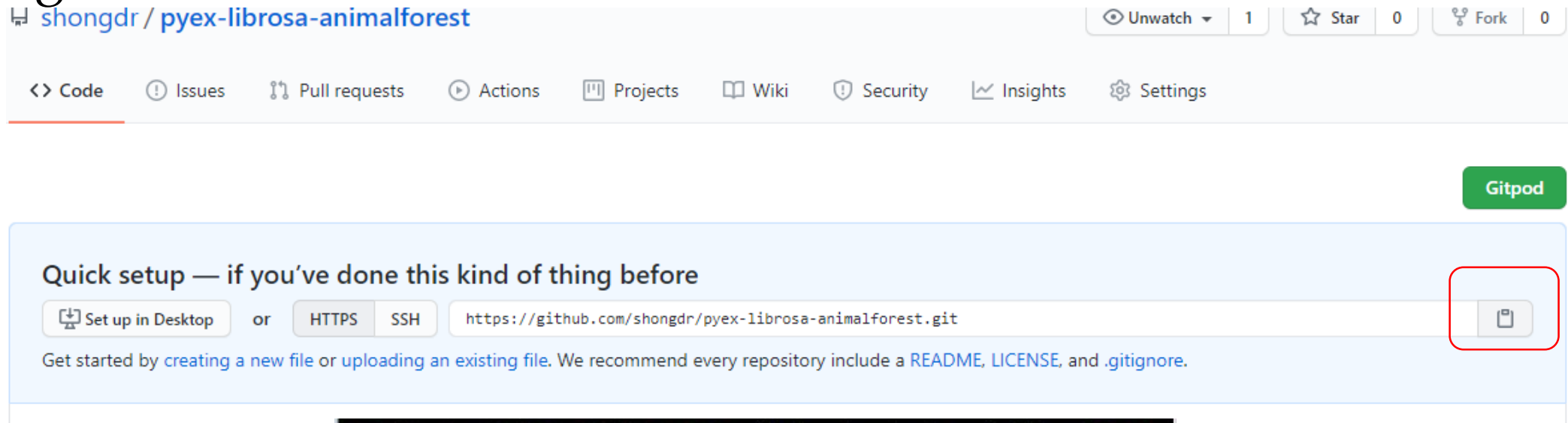
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   README.md
    new file:   aniforest-librosa-pitch-lect.ipynb
    new file:   aniforest-librosa-pitch.ipynb
    new file:   aniforest_librosa.ipynb
```

git commit 소스를 기여할 때, 기록할 말

\$ git commit -m "[Push 메시지명]"

git 업로드 할 주소 복사해오기 → 마우스 오른쪽 버튼 Paste



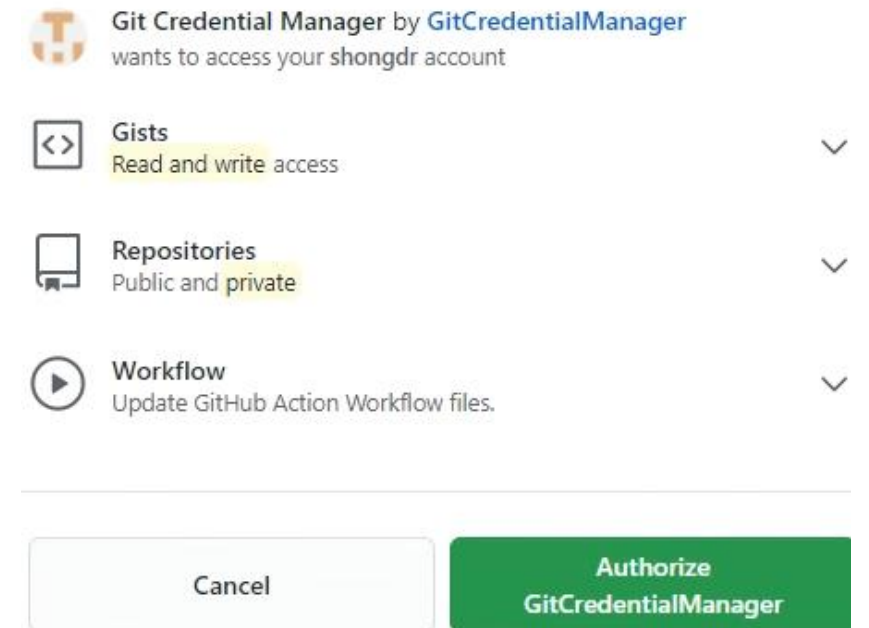
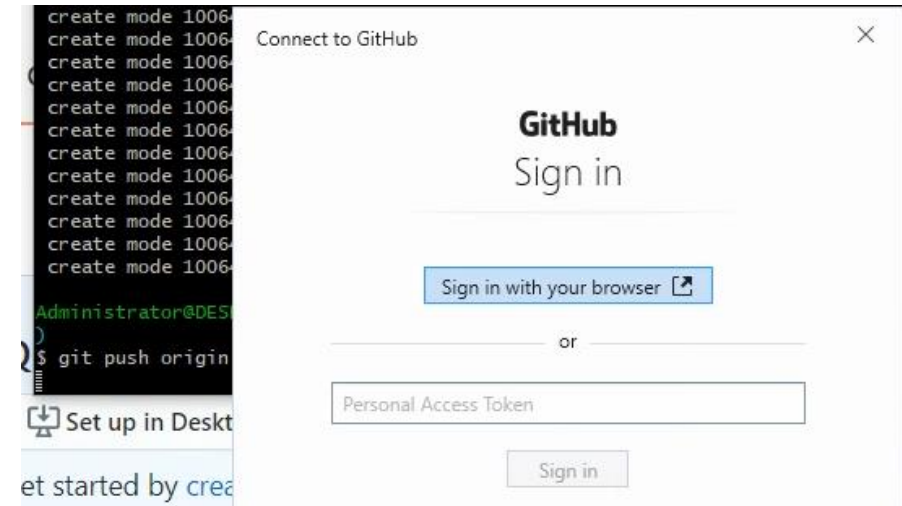
git init → git remote

git 리모트를 설정

(git 리모트란 git을 원격저장소에 저장하는 곳 의미)

\$ git remote add origin

<https://github.com/shongdr/doumcode.git>



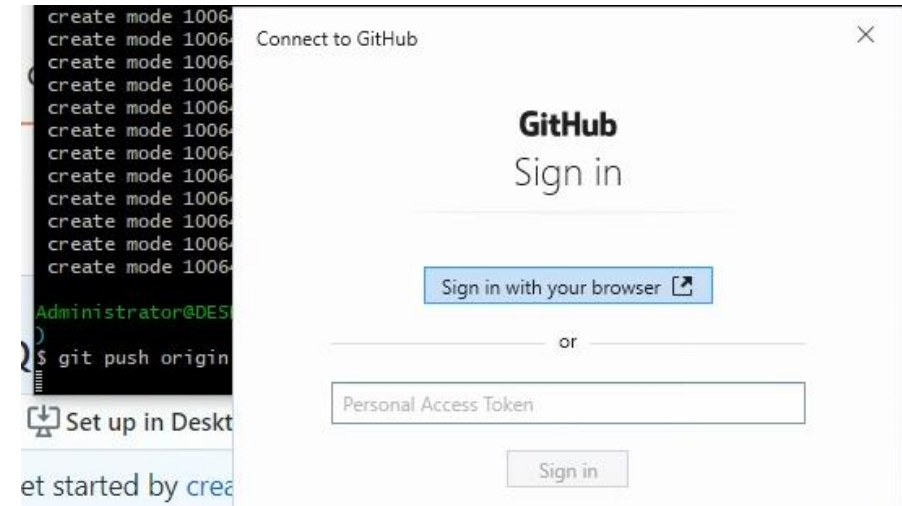
git init → git remote

\$ git remote add origin


<https://github.com/shongdr/doumcode.git>

git 리모트 URL (원격저장소)에 저장된 파일을 복사해올 수 있습니다.

이때 git clone 명령어를 사용하여 클론합니다.



 Git Credential Manager by GitCredentialManager wants to access your shongdr account

 Gists
Read and write access

 Repositories
Public and private

 Workflow
Update GitHub Action Workflow files.

Cancel

Authorize
GitCredentialManager

```
Administrator@DESKTOP-61J55V0 MINGW64 ~/lect_py/pyex/voice-animalforest (master)
$ git push origin master
Enumerating objects: 315, done.
Counting objects: 100% (315/315), done.
Delta compression using up to 24 threads
Compressing objects: 99% (311/313)
```

SSH

git 연결을 보다 안전하고 빠르게 하기 위해서는 SSH Key 등록을 권장.
이미 존재하는 문서로 SSH 생성 가이드를 참고하시거나 아래 절차를
따라주시면 됩니다.

우선 ssh-keygen 명령어로 SSH Key를 생성하시면 됩니다.

SSH Key를 생성하셨으면 ~/[사용자 폴더]/.ssh/ 에 파일이 존재하는
것을 확인하실 수 있습니다.

SSH 인증

생성한 키 중 `id_rsa.pub`는 GitHub에 등록해주셔야 합니다.

아래 절차를 따라해주시면 됩니다.

GitHub 홈페이지를 접속하셔서 로그인을 해주세요.

Profile 중 Settings 메뉴를 눌러주세요 (아래 그림을 참고해주세요.)

SSH 인증

Q. SSH 설정을 해도 아이디와 비밀번호를 물어봐요!

접속 정보에서 Use SSH를 클릭해 SSH 접속 정보를 이용하시기 바랍니다.

이때, git remote set-url 명령어를 이용하여

기존의 원격지 주소를 수정해야 합니다.

혹시 HTTPS 주소를 Remote URL로 사용하는지 체크해주세요

Remote URL은 ssh 포맷을 사용해주셔야 ssh 인증을 통해

입력을 넘어가실 수 있습니다.

origin의 Remote URL 변경방법.

\$ git remote set-url origin git@github.com:shongdr/doumcode.git

- 여러분이 퇴근길에 페이스북으로 글을 둘러보며 좋아요 하듯이 개발자들은 깃헙으로 스타(star)를 날립니다.

쉽게 생각해서

- Microsoft® Office를 Office 365로
- 서비스하는 것과 비슷하다 생각해주세요.

- # origin의 Remote URL이 제대로 변경됐는지 체크해주세요.

```
$ git remote show origin
```

■ 소스 기록

소스를 업로드 하기 위해서는 git add 명령어를 이용합니다.

샘플을 참고하세요

```
$ git add .
```

ignore 파일이나, 삭제한 파일 이력까지 커밋을 하실 경우, -f 옵션을 이용합니다.

```
$ git add . -f
```

git remote show origin을 통해 origin에 리모트 주소가 잘 등록되었는지 확인해보세요.

□ 소스 커밋

소스를 커밋하시면 staged 상태의 파일이 히스토리로 기록되고 적재됩니다.
파일 추적상태의 경우 git status 명령을 이용해서 확인합니다.

```
$ git status
```

git add 이후 git status를 하면 아래와 같은 화면이 나옵니다.

Staged 상태의 파일은 아직 기록된 상태가 아닙니다.

파일의 기록을 위해서는 커밋 작업이 필요합니다.

git commit 명령을 통해 Staged 상태의 파일을 커밋할 수 있습니다.

-m 옵션을 이용하여 커밋 메시지를 작성하는 것을 권장합니다.

실수로 커밋을 하여, 다시 커밋을 할 경우 커밋을 덮어씌울 수 있습니다. 이때 --amend 옵션을 이용합니다.

```
$ git add *
```

```
$ git commit -m "UI 레이아웃 이슈 수정."
```

- # 수정사항 발생

```
$ git add *
```

```
$ git commit -m "UI 레이아웃 이슈 수정 및 관리자 벨리데이션 추가." --amend
```

🔗 소스 업데이트

상대방이 커밋한 파일은 명령어를 통해서 직접 업데이트를 하셔야 동기화가 됩니다.

이때 사용하는 명령어는 git pull과 git fetch가 있습니다.

master 브랜치를 pull하여 업데이트

```
$ git pull origin master
```


- # master 브랜치를 fetch하여 업데이트

```
$ git fetch origin master
```

pull 과 fetch 의 차이점은 merge 작업을 하느냐 안하느냐로 나뉘어지며.

pull 은 fetch + merge 작업이라고 생각하시면 됩니다.

④ 소스 복원

여러분이 git을 쓰는 이유중에 중요한 부분을 차지하는 영역입니다.

정상적으로 커밋된 히스토리는, 리비전으로 git에 관리됩니다.

실수로 잘못 작업하였거나, 예전 버전으로 롤백하여 적용할 경우 여러분은 예전 버전으로 리셋하실 수 있습니다.

리셋은 git reset 명령을 사용합니다.

```
$ git reset HEAD^ --soft
```

git reset 다음 인수로는 되돌리는 버전의 위치를 가리킵니다.

- 현재위치(HEAD)를 기준하여 상대적인 위치를 설정하거나, 특정 버전 리비전 고유의 해시값을 지정합니다.

HEAD를 확인하시고 싶으면 `git reflog` 명령을 이용합니다.

`git reset`의 옵션 중 리셋 특성을 정하는 `--soft`, `--hard`, `--mixed` 옵션이 있습니다.

위 옵션은 아래에서 자세히 설명합니다.

`--soft`는 약한특성의 리셋입니다, 되돌릴 때 기존의 인덱스와 워킹트리를 보존합니다.

`--hard`는 강한특성의 리셋입니다, 되돌릴 때 기존의 인덱스와 워킹트리를 버립니다.

`--mixed`는 중간특성의 리셋입니다, 되돌릴 때 기존의 인덱스는 버리고 워킹트리를 보존합니다.

되돌리는 위치의 경우 아래와 같은 타입이 있습니다.

- # 바로 이전 단계로 인덱스와 워킹트리를 버리고 리셋.

```
$ git reset HEAD^ --hard
```

바로 두번째 전 단계로 인덱스와 워킹트리를 버리고 리셋.

```
$ git reset HEAD~2 --hard
```

특정 리비전의 기록으로 인덱스는 버리고 워킹트리를 보존하여 리셋.

```
$ git reset 991ee8c --mixed
```

- 🌿 브랜치

브랜치는 한국말로 가지(branch)입니다.

git에서는 마치 가지를 펼치듯 하나의 근본에서 여러 갈래로 쪼개어 관리할 수 있습니다.

이미지 출처 StackOverflow

branch의 특징은 아래와 같습니다.

기본은 master 브랜치라고 불리며, 필수로 제공되는 브랜치이다.

서로다른 브랜치들은 같은 조상을 가지고 있다.

브랜치를 새로 만드신다면 git branch [브랜치명]으로 생성합니다.

아래 명령라인에서는 new라는 브랜치를 생성하고 있습니다.

```
$ git branch new
```

master 기준으로 new를 브랜치(가지치기)하면 master와 똑같은 소스코드가 new에도 적용됩니다.

- 하지만 이 이후로 new에서 코드를 수정하면, master와 new는 서로 다른 코드가 되기 때문에 갈라집니다.

생성된 new 브랜치로 접속하기 위해서는 git checkout [브랜치명]을 이용합니다.

```
$ git checkout new
```

생성과정과 브랜치 이동과정을 동시에 하고자 하면 git checkout에 -b 옵션을 이용합니다.

브랜치 new를 생성과 동시에 체크아웃.

```
$ git checkout -b new
```

생성한 브랜치는 현재 로컬에 저장되어 있습니다.

협업 작업에서는 생성한 브랜치를 원격 저장소에 등록해주어야 합니다.

이때는 git push [브랜치명]을 이용합니다.

```
$ git push origin new
```

브랜치 생성 및 등록의 과정은 아래 화면과 같습니다.

- 브랜치의 삭제는 git branch 명령에서 -d 옵션을 사용합니다.

삭제된 브랜치 또한 원격 저장소에 반영을 해야합니다.

이때 브랜치 명 앞에 콜론(:)을 붙여주어야 하니 이 점 주의해주세요.

☹ 소스 병합

브랜치를 사용하는 과정에서 가장 중요한 머지와 리베이스 등의 병합 기법입니다.

서로 다른 브랜치에서 서로 다른 코드가 개발되었고, 실제 배포에서 이를 합쳐야 할 때 사용합니다.

병합 방식에서는 크게 git merge와 git rebase가 존재합니다.

머지 방식과 리베이스 방식의 차이는 아래 이미지를 확인해주세요.

이미지 출처 <http://git.mikeward.org/>

- 아래는 머지해야 하는 상황을 구현해봤습니다.

master에서 sub branch가 생성되었으며, master 브랜치에서 sub 브랜치를 머지하고자 합니다.

파일 구성은 아래와 같습니다.

* master -> some_file.txt의 내용

* 1번째 단계 HEAD

I'm a file.

* sub -> some_file.txt의 내용

* 2번째 단계 HEAD (최신)

I'm a file.

- Inserted new line from the sub branch.

```
$ git checkout -f master
```

```
$ git merge sub
```

```
# 현재 브랜치 master, 대상 브랜치 sub.
```

```
# master에서 sub를 머지합니다.
```

```
# HEAD -> master
```

```
# sub -> sub
```

머지 이후 master에서 파일을 보면, 아래와 같은 내용을 얻습니다.

```
* merge 이후 master -> some_file.txt
```

```
I'm a file.
```

Inserted new line from the sub branch.

- ☺ 충돌과 해결

git으로 merge, rebase 수행시 충돌(conflict)가 발생 할 수 있습니다.

이는 같은 조상을 기준으로, 서로 다른 두개의 브랜치가 같은 소스코드를 변경했을 때 발생합니다.

* master -> some_file.txt의 내용

Apple

위는 master 브랜치의 some_file.txt의 내용이다.

아래는 해당 브랜치를 복제한 sub 브랜치이며, 복제 이후 한번 내용을 수정하였습니다.

* sub -> some_file.txt의 내용

* 2번째 단계 HEAD

Banana

이후 master에서도 내용을 변경하여 버전을 업데이트 합니다.

* master -> some_file.txt의 내용

* 2번째 단계 HEAD(sub랑 단계가 겹침)

Strawberry

둘 모두 버전이 같으나 같은 라인에서 변경사항이 발생했습니다.

이 경우 충돌이 발생합니다.

충돌이 발생한 some_file.txt를 열어보면 아래와 같은 내용을 보실 수 있습니다.

* 머지 이후 master -> some_file.txt (충돌)

<<<<<< HEAD

Strawberry

=====

Banana

>>>>>> sub

여기서 HEAD는 현재 브랜치(master)를 의미합니다.

HEAD와 sub의 각각 내용을 보여주고 있는데 꺾쇠(<, >), 이퀄(=)기호가 없도록 문장 하나를 선택해서 반영해주어야

충돌이 해결 될 수 있습니다.

여기서는 master 브랜치의 Strawberry를 선택하여 충돌을 해결하겠습니다.

* 머지 이후 master -> some_file.txt (수정)

Strawberry

수정이 되었다면 머지 해결을 위해 git add와 git commit으로 충돌(conflict)을 해결하세요.

\$ git add *

\$ git commit -m "Solved the conflict issue."