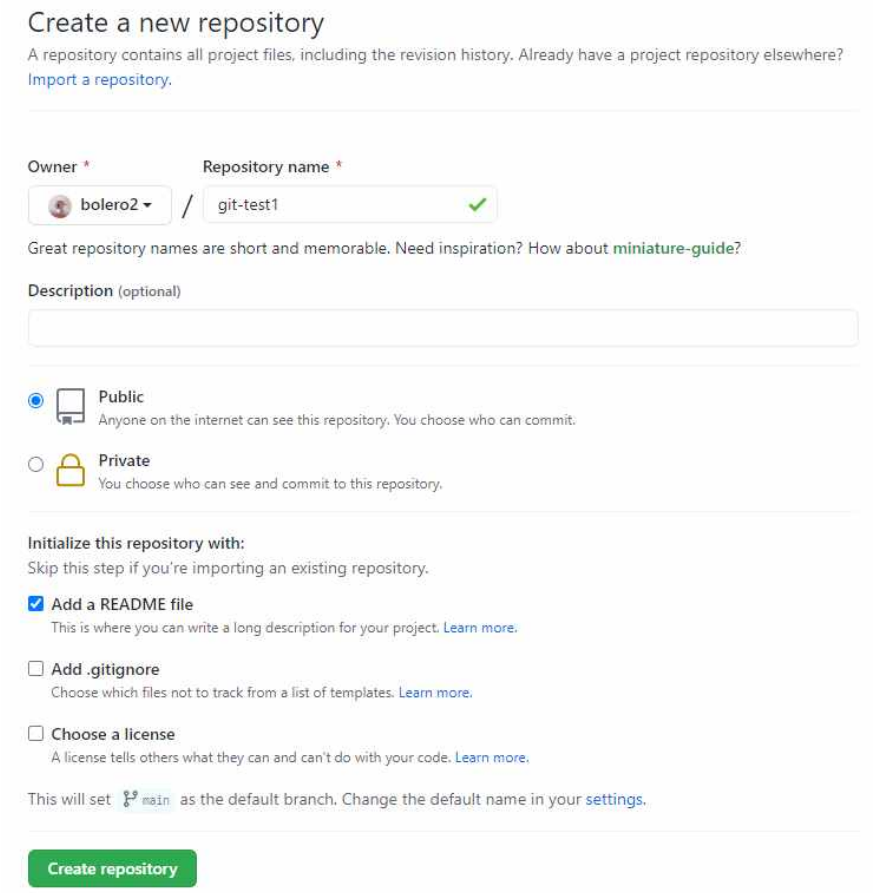
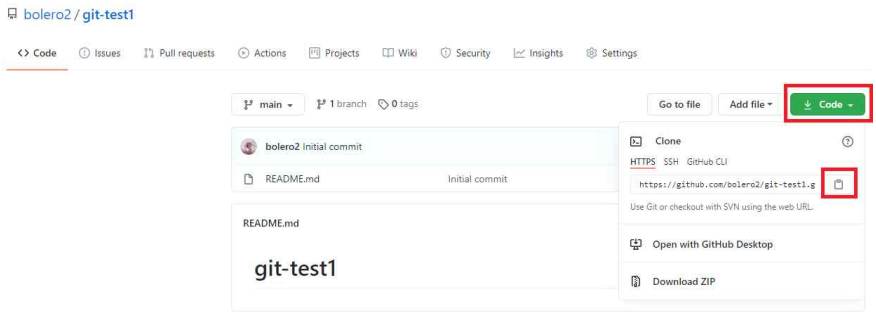
1. Repository 만들기

1) github.com 사이트에서 Repository 생성함.



2) terminal에서 git clone 실행




```
(base) clt_dc@DCGPU:~/git$ git clone https://github.com/bolero2/git-test1.git
Cloning into 'git-test1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

```
(base) clt_dc@DCGPU:~/git$ ls
git-test1
```

로컬 저장소에 중앙 서버의 Repository가 저장됨.

(해당 원격 저장소가 이전부터 Local PC에 있었으면, git clone 하지 않아도 됨. 최초 1회용)

2. Git 파일/작업 추가하기

(대부분, git clone으로 저장소를 저장하는 것으로 시작함.)

[용어 정리]

- 원격 저장소 : github.com에서 확인 가능한 저장소
- 로컬 저장소 : Local PC에 저장된(git clone 등으로 불러온) git 작업 공간

1) git status : 현재 상태 확인

```
(base) clt_dc@DCGPU:~/git/git-test1$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

origin : 원격 저장소 이름
main : Branch 이름
이 두개는 기억해야 함.

```
(base) clt_dc@DCGPU:~/git/git-test1$ ls
README.md
```

현재 README.md 밖에 없음.

2) git pull : 로컬 저장소의 작업 공간 최신화

▶ 원격 저장소에 있는 것들을 불러와서 로컬 저장소의 것들과 Merge 하는 작업. (변경사항을 모두 덮어쓴다는 뜻)

```
(base) clt_dc@DCGPU:~/git/git-test1$ git pull
Already up to date.
```

3) 본 작업 수행(이 작업은 로컬 저장소에서 진행됨, 원격 저장소와는 무관함.)

▶ .py 등 소스 코드 작업을 진행함.

```
(base) clt_dc@DCGPU:~/git/git-test1$ vi test1.py
(base) clt_dc@DCGPU:~/git/git-test1$ python test1.py
3.141592653589793
(base) clt_dc@DCGPU:~/git/git-test1$ vi test1.c
(base) clt_dc@DCGPU:~/git/git-test1$ gcc -o test1 test1.c
(base) clt_dc@DCGPU:~/git/git-test1$ ls
README.md  test1  test1.c  test1.py
(base) clt_dc@DCGPU:~/git/git-test1$ ./test1
A + B is 30
```

[저장소의 상태 표]

Tracked			Untracked
Unmodified	Modified	Staged	

- **Tracked : 관리 대상**
- **Untracked : 관리 대상이 아님**
- Unmodified : 수정하지 않음
- Modified : 수정함
- Staged : Commit 시에 중앙 저장소에 기록됨.

(처음 원격 저장소 git clone 시 모든 파일은 Tracked이면서 Unmodified 상태임.)

4) git status : 작업 완료 시 어떤 파일이 수정되었는지 확인.

```
(base) clt_dc@DCGPU:~/git/git-test1$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

        test1
        test1.c
        test1.py

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

- 위의 3개 파일은 Untracked 파일임.
- git clone 시에 내려온 파일이 아니기 때문.

5) git add : Untracked 파일을 Tracked 상태로 변경.

- git add test1.py test1.c test1
- git add ./*
- git add test*.py
- git add *.*

모두 가능함. (이후 **git status**로 상태 확인을 꼭 해줘야 한다.)

```
(base) clt_dc@DCGPU:~/git/git-test1$ git add test1.py test1.c
(base) clt_dc@DCGPU:~/git/git-test1$ git add ./*
(base) clt_dc@DCGPU:~/git/git-test1$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   test1
        new file:   test1.c
        new file:   test1.py
```

“new file:”로 넘어간 파일들은 이제 Tracked 상태임.

6) git commit –m “commit message 내용” : Staged 파일들을 모두 commit 함. (변경사항 기록)

```
(base) clt_dc@DCGPU:~/git/git-test1$ git commit -m "first commit"
[main cdd495a] first commit
 Committer: Ubuntu <clt_dc@DCGPU.arkhpeawxyoutnzh1vellybbwc.cx.internal.cloudapp.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 3 files changed, 13 insertions(+)
 create mode 100755 test1
 create mode 100644 test1.c
 create mode 100644 test1.py
```

(최종)

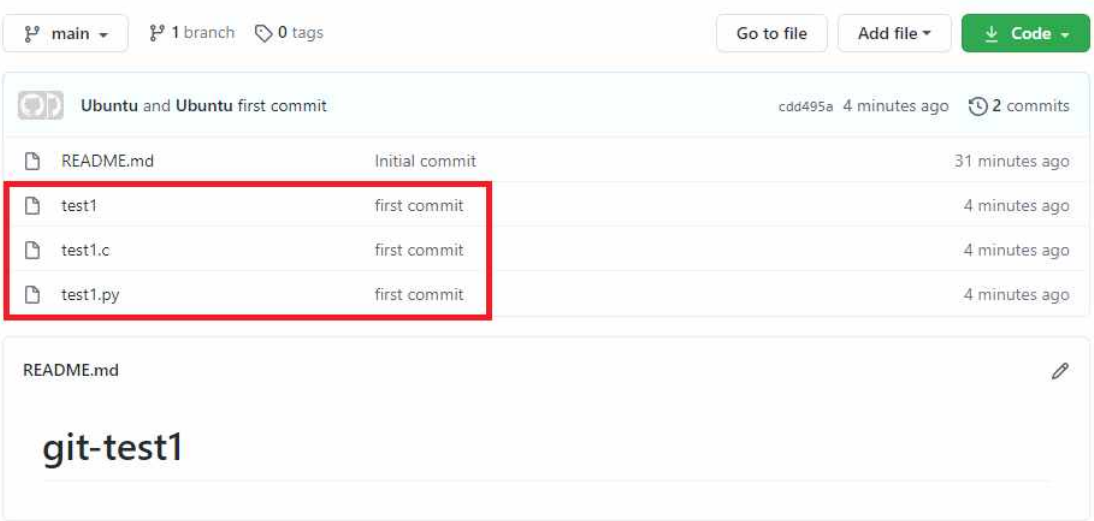
7) git push -u origin main : 로컬 저장소의 commit 내용을 원격 저장소로 전송.

```
(base) clt_dc@DCGPU:~/git/git-test1$ git push -u origin main
Username for 'https://github.com': bolero2
Password for 'https://bolero2@github.com':
Counting objects: 5, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.82 KiB | 2.82 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/bolero2/git-test1.git
   b4da483..cdd495a  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- origin : 원격 저장소
- main : branch 이름
- -u : update 옵션.

(git clone이 여러 개 있을 때, 다른 사람이 push 한 후에 내가 push를 하면 동작 안 함. update로 merge 후 push 가능.)

8) 원격 저장소에서 확인하기



[기본 개념]

clone(최초 1회) → **pull(작업 전 최신화)** → 코드 작업

→ **add(Stage로 올림)** → **commit(변경사항 기록)** → **push(원격 저장소에 저장)**

3. Git 파일 삭제

(원격 저장소를 git clone으로 불러온 후, git pull까지 완료한 상태.)

1) git rm

- -r : Recursive 삭제.(directory 하위를 순회하며 모두 삭제)
- -f : 변경사항을 commit 하지 않았을 경우 강제로 삭제.

[현재 상태]

```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ ls
README.md  data-handling-scripts  mobilenet-v2-tensorflow  shufflenet-v1-tensorflow

utility  utils.old  vgg16-tensorflow  yolo-v1-tensorflow  yolo-v3-tensorflow
```

[git rm -rf 명령어 실행]

- git rm -rf ./*-tensorflow
- git rm -rf ./utility

: 파일 하나 단위로도 수행 가능. (ex. git rm test1.py)

```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ git rm -rf ./*-tensorflow
rm 'mobilenet-v2-tensorflow/mobilenet-v2.py'
rm 'shufflenet-v1-tensorflow/shufflenet-v1.py'
rm 'vgg16-tensorflow/vgg16-conv3d.py'
rm 'yolo-v1-tensorflow/README.md'
rm 'yolo-v1-tensorflow/airplane.jpg'
rm 'yolo-v1-tensorflow/config.py'
rm 'yolo-v1-tensorflow/cvtest.py'
rm 'yolo-v1-tensorflow/hrnet-w32.py'
rm 'yolo-v1-tensorflow/main_ssd_compare_originalVGGnet19.py'
rm 'yolo-v1-tensorflow/sheep.jpg'
rm 'yolo-v1-tensorflow/yolo-v1-main.py'
rm 'yolo-v1-tensorflow/yolo-v1-network.py'
rm 'yolo-v1-tensorflow/yolo-v1-test.py'
rm 'yolo-v1-tensorflow/yolo-v1-total-colab.py'
rm 'yolo-v1-tensorflow/yolo-v1-total.py'
rm 'yolo-v1-tensorflow/yolo-v1-total2-temp.py'
rm 'yolo-v3-tensorflow/class_darknet53.py'
rm 'yolo-v3-tensorflow/class_detection.py'
rm 'yolo-v3-tensorflow/class_utils.py'
rm 'yolo-v3-tensorflow/class_yolo.py'
rm 'yolo-v3-tensorflow/yolo-dc.py'
```

```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ git rm -rf ./utility
rm 'utility/3D_NUMBER_Demo_1.py'
rm 'utility/Image_Augmentation.py'
rm 'utility/python-csv.py'
rm 'utility/test_pytorch.py'
```

[git status : 현재 stage 상태 확인]

```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    mobilenet-v2-tensorflow/mobilenet-v2.py
        deleted:    shufflenet-v1-tensorflow/shufflenet-v1.py
        deleted:    utility/3D_NUMBER_Demo_1.py
        deleted:    utility/Image_Augmentation.py
        deleted:    utility/python-csv.py
        deleted:    utility/test_pytorch.py
        deleted:    vgg16-tensorflow/vgg16-conv3d.py
        deleted:    yolo-v1-tensorflow/README.md
        deleted:    yolo-v1-tensorflow/airplane.jpg
        deleted:    yolo-v1-tensorflow/config.py
        deleted:    yolo-v1-tensorflow/cvtest.py
        deleted:    yolo-v1-tensorflow/hrnet-w32.py
        deleted:    yolo-v1-tensorflow/main_ssd_compare_originalVGGnet19.py
        deleted:    yolo-v1-tensorflow/sheep.jpg
        deleted:    yolo-v1-tensorflow/yolo-v1-main.py
        deleted:    yolo-v1-tensorflow/yolo-v1-network.py
        deleted:    yolo-v1-tensorflow/yolo-v1-test.py
        deleted:    yolo-v1-tensorflow/yolo-v1-total-colab.py
        deleted:    yolo-v1-tensorflow/yolo-v1-total.py
        deleted:    yolo-v1-tensorflow/yolo-v1-total2-temp.py
        deleted:    yolo-v3-tensorflow/class_darknet53.py
        deleted:    yolo-v3-tensorflow/class_detection.py
        deleted:    yolo-v3-tensorflow/class_utils.py
        deleted:    yolo-v3-tensorflow/class_yolo.py
        deleted:    yolo-v3-tensorflow/yolo-dc.py
```

: 초록색으로 올라간 것은 stage 상태임을 뜻함.

[git commit –m “message” 명령어 실행]

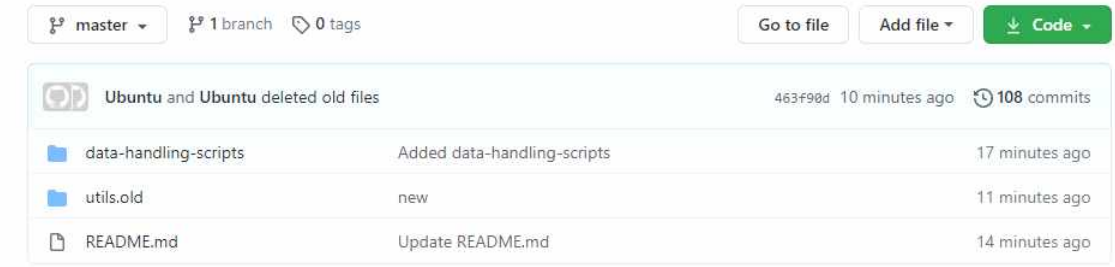
```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ git commit -m "deleted old files"
[master 463f90d] deleted old files
 Committer: Ubuntu <clt_dc@DCGPU.arkhpeawxyoutnzh1vellybbwc.cx.internal.cloudapp.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 25 files changed, 6083 deletions(-)
 delete mode 100644 mobilenet-v2-tensorflow/mobilenet-v2.py
 delete mode 100644 shufflenet-v1-tensorflow/shufflenet-v1.py
 delete mode 100644 utility/3D_NUMBER_Demo_1.py
 delete mode 100644 utility/Image_Augmentation.py
 delete mode 100644 utility/python-csv.py
 delete mode 100644 utility/test_pytorch.py
 delete mode 100644 vgg16-tensorflow/vgg16-conv3d.py
 delete mode 100644 yolo-v1-tensorflow/README.md
 delete mode 100644 yolo-v1-tensorflow/airplane.jpg
 delete mode 100644 yolo-v1-tensorflow/config.py
 delete mode 100644 yolo-v1-tensorflow/cvtest.py
 delete mode 100644 yolo-v1-tensorflow/hrnet-w32.py
 delete mode 100644 yolo-v1-tensorflow/main_ssd_compare_originalVGGnet19.py
 delete mode 100644 yolo-v1-tensorflow/sheep.jpg
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-main.py
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-network.py
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-test.py
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-total-colab.py
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-total.py
 delete mode 100644 yolo-v1-tensorflow/yolo-v1-total2-temp.py
 delete mode 100644 yolo-v3-tensorflow/class_darknet53.py
 delete mode 100644 yolo-v3-tensorflow/class_detection.py
 delete mode 100644 yolo-v3-tensorflow/class_utils.py
 delete mode 100644 yolo-v3-tensorflow/class_yolo.py
 delete mode 100644 yolo-v3-tensorflow/yolo-dc.py
```

:25 files changed, 6083 deletions(-)를 보고, 잘 삭제되었음을 알 수 있음.

[git push –u origin master 명령어 실행]
: 최종 단계, commit 내용을 원격 저장소로 올림]

```
(base) clt_dc@DCGPU:~/git/DeepLearning-dc$ git push -u origin master
Username for 'https://github.com': bolero2
Password for 'https://bolero2@github.com':
Counting objects: 2, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 264 bytes | 264.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bolero2/DeepLearning-dc.git
   4b360aa..463f90d  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

2) 원격 저장소 확인



: 잘 삭제되었음을 확인할 수 있다.

My Memo

[Git 사용법]

1. 로컬 저장소 만들기 in laptop
> git init

2. 현재 로컬 저장소 상태 확인

중앙 서버 -> github 사이트
local pc -> 내 로컬 저장소

[Repository 만들기]
항상 중앙 서버에서(github에서) 레포를 하나 만들고 clone으로 시작하는 것이 가장 편함.

1. git add *.py
모든 py 파일을 stage 상태로 넘김

2. git commit -m "commit 메세지"

3. git push -u origin main

[git 사용법]
1. Git Clone 실행하기
-> git 중앙 서버의 repository를 나의 local pc에 복사하는 것을 의미,
해당 repo가 이전부터 local pc에 있었으면, clone 하지 않아도 됨.

2. git pull
-> 파일의 최신화하는 작업.
만약 git clone한 것이 아닐경우(이 경우가 대부분의 경우임)
중앙 서버에서 해당 저장소의 파일들을 불러와서, 현재 파일과 merge 함.(변경사항을 모두 덮어쓴다는 뜻)

3. 파일 수정
-> 현재 로컬 저장소에 있는 소스코드를 수정하는 작업.

로컬 저장소 ->
1) Tracked(관리 대상) / 2) Untracked(관리 대상이 아님)
Tracked -> unmodified / modified / staged
unmodified : 수정하지 않음
modified : 수정함
staged : commit시 중앙 서버(저장소)에 기록됨.

(처음 저장소 Clone시 모든 파일은 Tracked이면서 Unmodified 상태.)

4. git status
수정내역을 보여줌. On branch 뒤에 있는 내용이 branch 이름임.
대부분의 경우 master 혹은 main으로 되어 있을 듯.

5. git add *
(git add *.*, git add ./*, git add test.py 등등...)
Untracked 파일들을 모두 stage로 올림. tracked 시키기 위함.
이후 commit시 완료(아직은 로컬에 있는 상태임)

6. git status
git add로 올린 파일이 잘 들어갔는지 확인(변경사항 확인)

7. git commit -m "commit message 내용"

(최종) 8. git push -u origin main
origin은 원격 저장소
main은 branch(master든 main이든...)
-u 옵션은 update 옵션임. Clone 사람이 여러명이 있을때, 다른 사람이 push 한 후에 push를 하면
작동이 안됨. 그래서 update로 merge 후 push 가능

만약 origin(원격 저장소) 이 -> git clone으로 받은 것이 아니라면?
git remote add origin (원격 저장소 주소)
를 해서 원격 저장소 위치를 알려줘야 함.

[Source Tree]
소스트리는 위의 과정을 GUI로 보여줌.
항상 clone -> pull -> commit -> push 과정을 거치는 것으로 동일함.