18기 지능팀 예비단원 윤형준 리눅스&git 보고서

1. 우분투

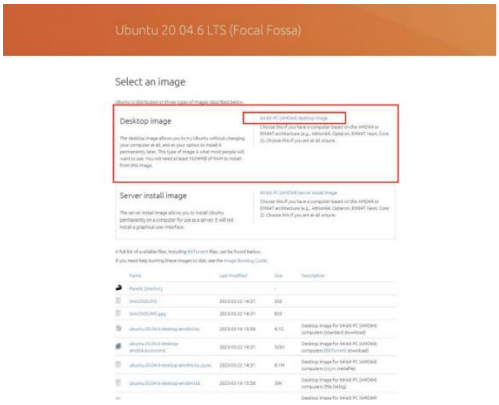
우분투란 간단히 컴퓨터 운영체제이다. 풀어서 말하면 컴퓨터에서 프로그램과 주변기기를 사용할 수 있도록 해주는 운영체제 중 하나이다. 안드로이드 운영체제처럼 리눅스 커널에 기반한 운영체제로 모바일과 데스크톱PC, 서버에도 우분투 운영체제를 설치해 사용할 수 있다.

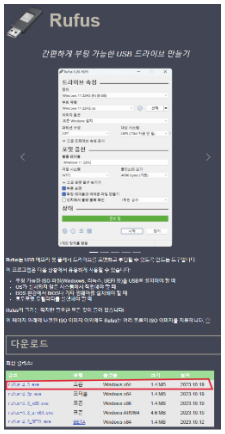
우분투와 리눅스의 차이는 무엇인가? 이를 이해하기 위해서는 커널(kernel)과 배포판이라는 것에 대해 알아야 한다. 커널(kernel)이란 컴퓨터의 운영체제의 핵심이 되는 컴퓨터 프로그램의 하나이다. 시스템의 모든 것을 완전히 통제하고 운영체제의 다른 부분 및 응용 프로그램 수행에 필요한 여러 가지 서비스를 제공한다. 배포판이란 여러 종류의 프로그램을 꾸러미 하나로 모아놓은 것을 말한다. 이제 우분투와 리눅스의 차이를 알아보자면, 리눅스는 커널이고 우분투는 리눅스 배포판이다. 즉 리눅스 프로그램 중 상호작용이 잘되는 두 가지 이상의 프로그램을 모아 만든 것이 우분투이다.

2. 우분투(20.04) 설치

1) 우분투 설치 USB굽기

우분투 이미지 : https://releases.ubuntu.com/focal/ Rufus : https://rufus.ie/ko/



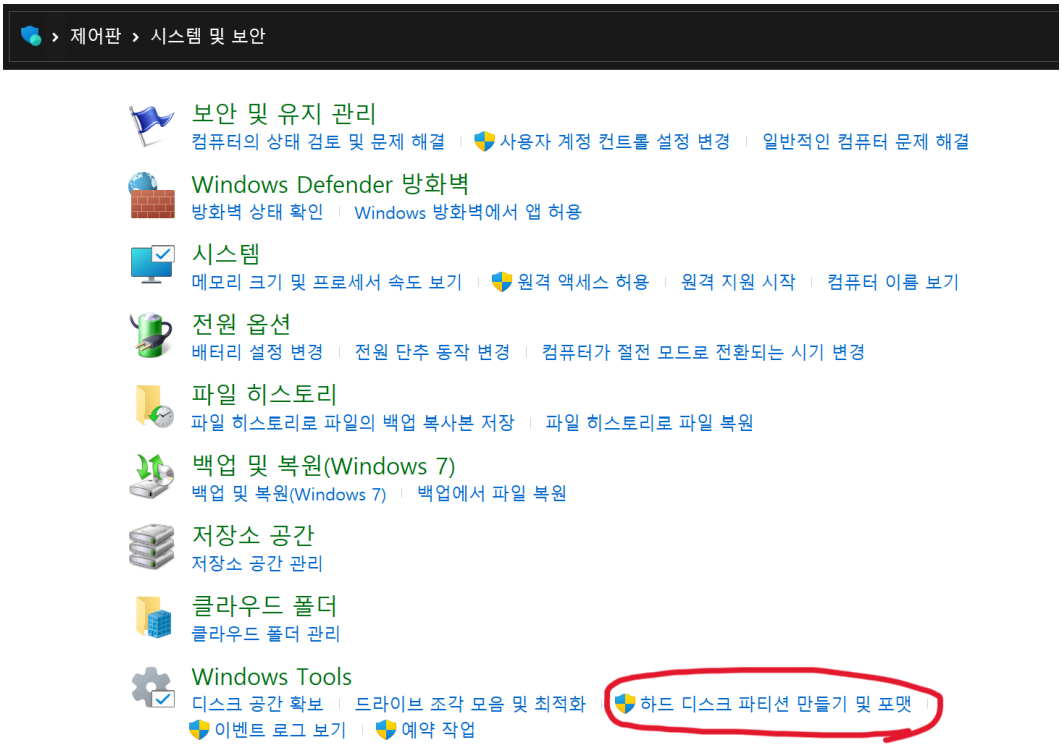


위 사진과 같이 우분투 이미지 파일을 다운로드 한 후 Rufus도 설치 후 실행. 빈 USB를 꽂고 장치에서 다중 파티션 선택 -> 부트 선택에서 디스크 또는 IOS 이미지 -> 선택 -> ubuntu 20.04 이미지 파일 ios 찾고 열기 -> 시작.

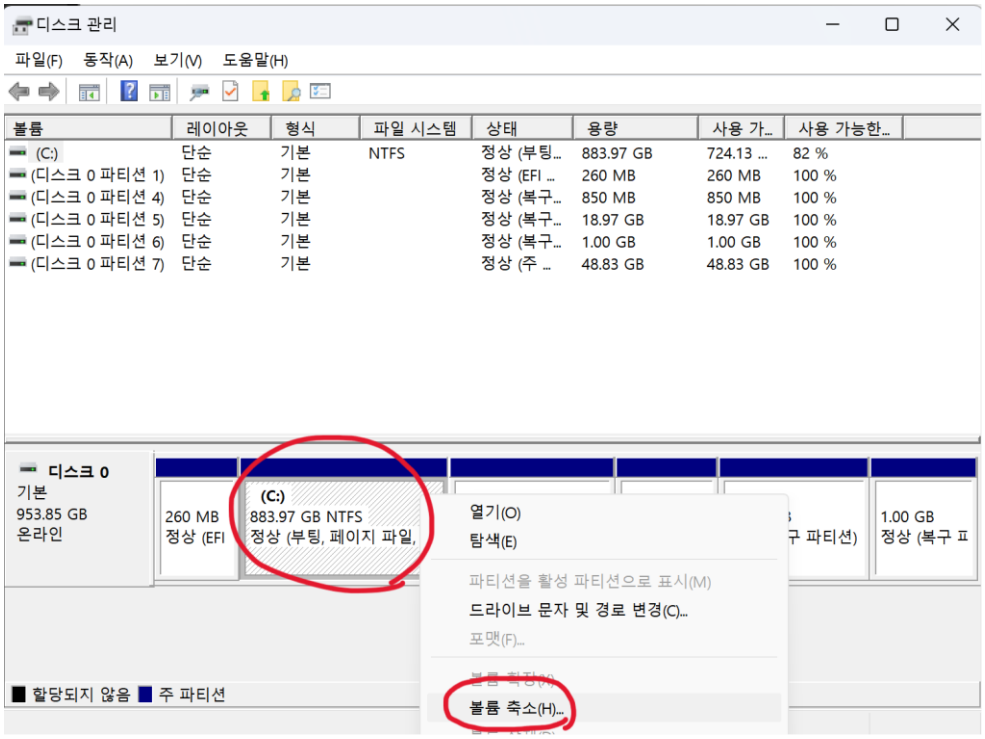
2) 설치

먼저 우분투 설치를 위한 용량을 할당한다.

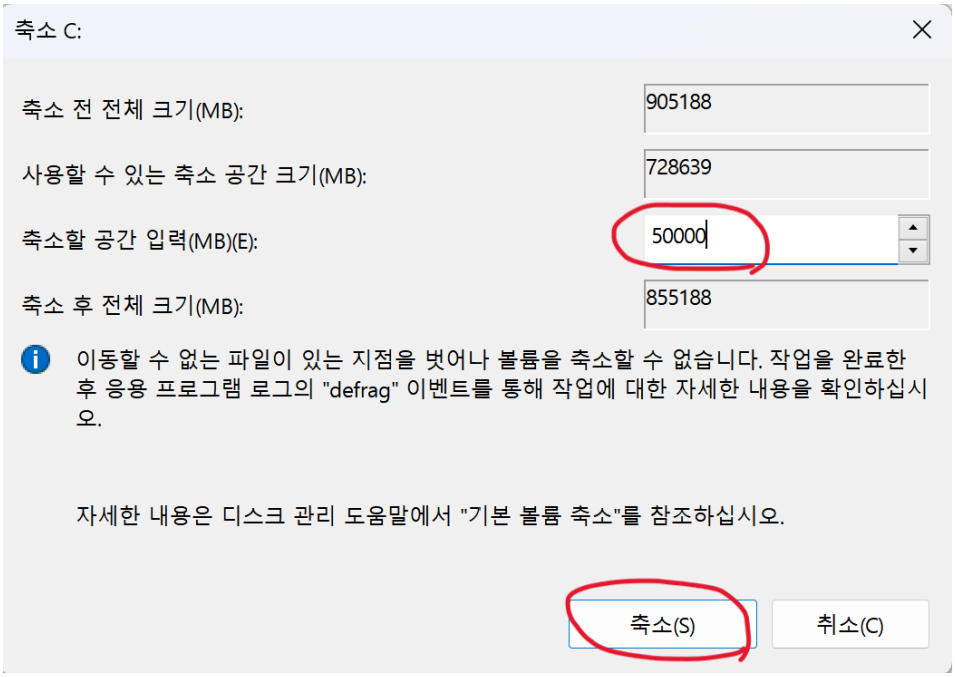
제어판 -> 시스템 및 보안 -> 하드 디스크 파티션 만들기 및 포맷



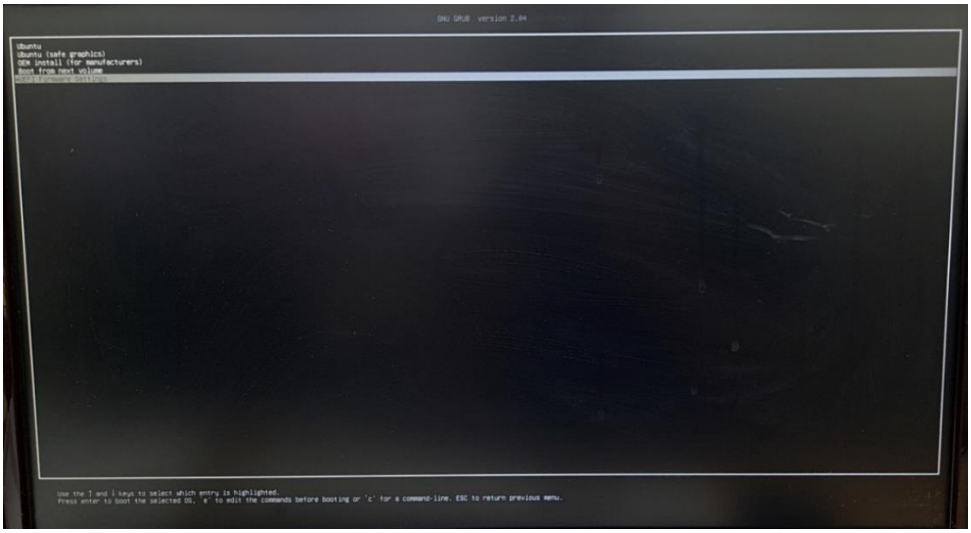
메인(C)디스크 우클릭 후 볼륨 축소.



컴퓨터의 사용 가능한 할당 크기를 봐가면서 30기가~60기가 중 할당.

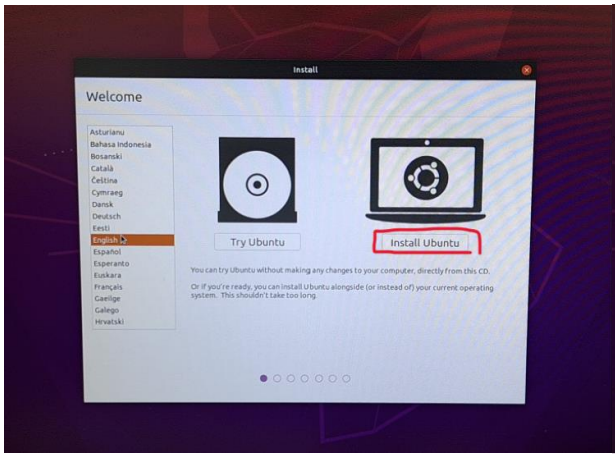


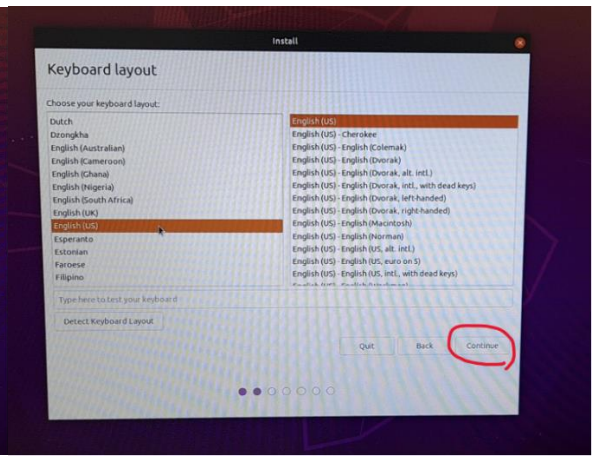
구워진 USB를 꽂고 BIOS 설정 진입(삼성은 F2연타) -> Boot Order Priorities에서 우분투 부팅 USB를 1순위로 변경 -> 저장 후 부팅.

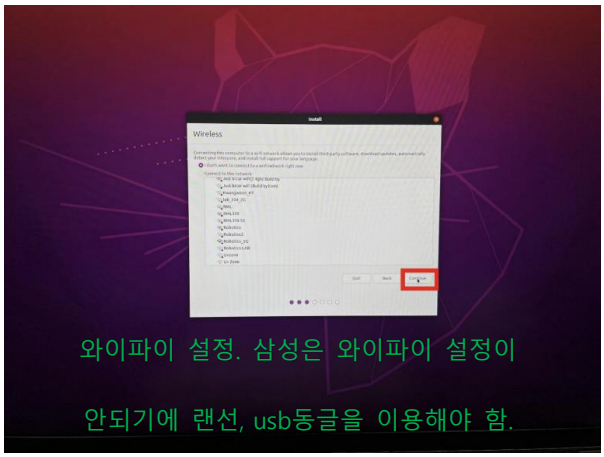


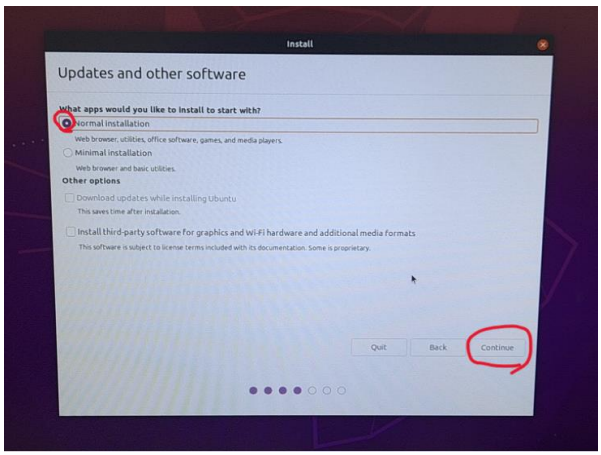
만약 위 사진에서 멈췄다면 UBUNTU(SAFE GRAPHICS) 선택.

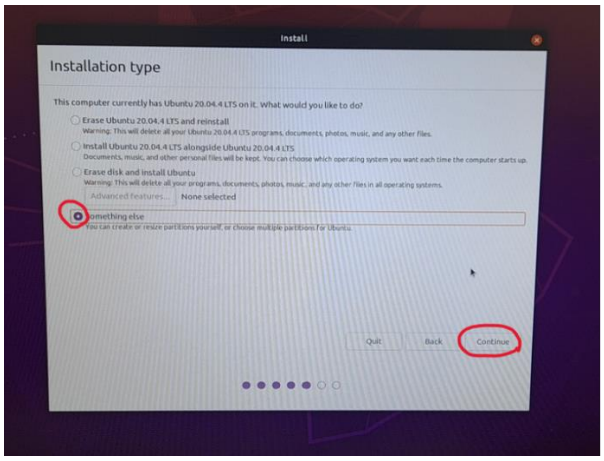
정상 작동하여 아래 사진들과 같이 뜬다면 사진을 따라한다. 설명이 쓰여 있지 않은 곳은 빨간색을 따라한다.

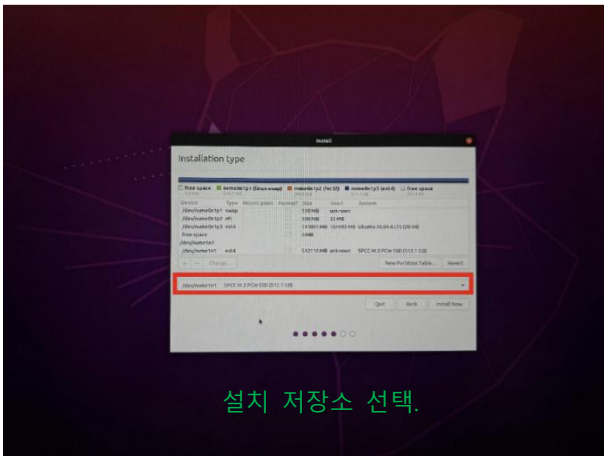


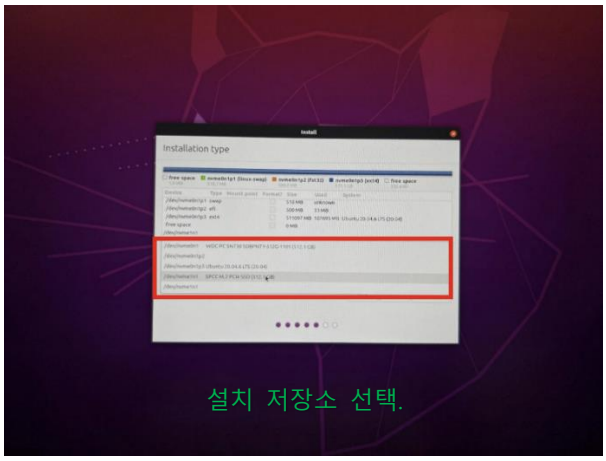


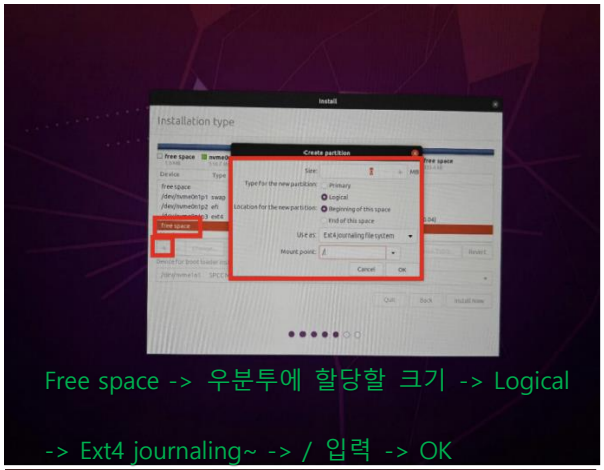


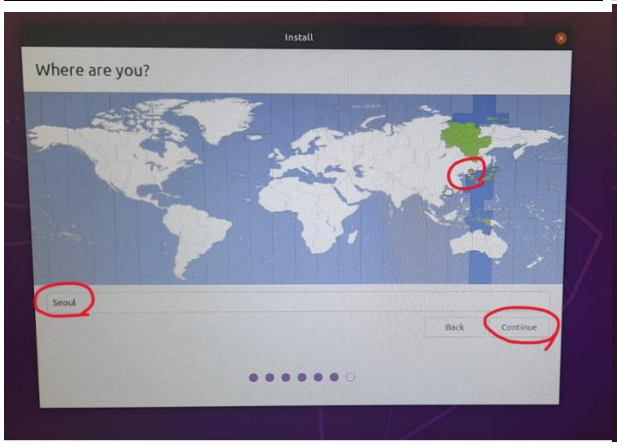


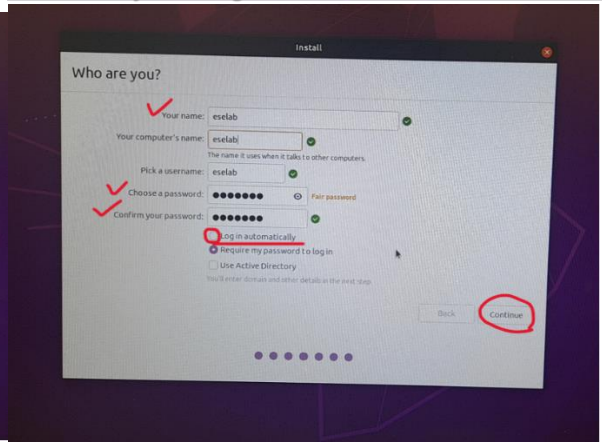












3. CLI (Command Line Interface) 명령어

명령어 인터페이스는 (마우스나 각종 UI 컴포넌트 대신) 터미널을 통해 사용자와 컴퓨터가 상호작용하는 방식

분류	명령어	설명
<관리자 권한으로 실행>	$ sudo (super user do)	1. 관리자만 읽을 수 있

		는 파일 읽기 2. 새로운 프로그램 설치 3. Notes : 새로운 프로그램 설치시 Package Manager를 이용하는 것이 보편적
<기본적인 unix/linux 명령어>	$ ls (list)	파일보기
	$ al (all)	파일의 세부내용 확인
	$ cd (change directory)	디렉토리로 이동
	$ pwd (print working directory)	Full 디렉토리 path 확인 / (디렉토리 위치 변경)
<디렉토리로 이동>	$ cd ~	홈 디렉토리 (현재 사용자 개인 파일 디렉토리)
	$ cd /	루트 디렉토리 (시스템 최상위 디렉토리 확인) / (매우 중요한 디렉토리를 의미하며 관리자 권한 요구)
	$ cd .	현재 디렉토리
	$ cd ..	부모 디렉토리
	$ pwd	현재 full 디렉토리 확인
	$ clear	터미널의 입력 내용들을 지워줌
<자주쓰는 명령어>	$ touch [file_name]	빈 파일 생성 / (not 디렉토리 / literally 파일 생성)
	$ mkdir [dir_name] (make directory)	디렉토리 생성
	$ cat [file_name] (concarenate)	텍스트 형태의 파일 확인
	$ mv [file_name or	파일 또는 디렉토리 옮기

	dic_name] [target_dir_name] (move)	기
	$ mv [file_name or dic_name] [target_file_name]	파일 및 디렉토리 바꾸기
	$ cp [file] [target_dir_name] (copy)	복사
	$ cp -r [folder_name]	복사
<삭제> (주의! 휴지통을 거치지 않고 즉시 삭제)	$ rm [file_name] (remove)	파일 삭제
	$ rm -r [dir_name]	폴더 삭제
<파일 소유권 변경>	$ chown [owner_file]: [group_file] (change owner)	파일 소유권 변경
	$ mkdir	디렉토리 생성
	$ rm	삭제
<GUI프로그램의 실행을 더 간편하게 해줌>	$ explorer .	현재 폴더를 windows파일 관리자에서 보기
	$ open .	현재 폴더를 macOS finder에서 보기
	$ code .	현재 폴더를 VS Code 에디터로 열기
<파일 이름에 따른 팁>	공백이 있는 경우	공백부분에 tab키를 누르기 or /(역슬래시) 입력
	파일 자동완성 기능	첫글자 등을 쓰고 tab키 누르기

4. GIT

1) Git이란

Git은 형상 관리 도구로 주로 여러명의 개발자가 하나의 소프트웨어 개발 프로젝트에 참여할 때, 소스 코드를 관리하는데 사용된다고 한다. 프로젝트를 진행하면서 소스코드를 USB나 메일로 주고 받는 건 엄청난 낭비임과 동시에 보안성 위험이 있다. 그렇기 때문에 프로젝트를 진행함에 있어서 형상 관리 도구를 사용한다.

형상 관리 도구(Git)의 장점 중 하나는 변경을 쉽게 되돌릴 수 있다는 것이다. 소스코드를 과거의 특정 시점으로 되돌리거나, 특정 시점의 변경사항을 취소하거나, 두 버전의 소스코드를 비교하는 등의 작업이 가능하다.

Github는 형상 관리 도구 웹 호스팅 시스템으로 Git을 사용하는 프로젝트를 지원한다. Git 사용 시 서버를 자체적으로 구축하는 것도 가능하지만, Github 서버를 이용하는 편이 편하다.

2) Git 설치

터미널 열기(단축키) : ctrl + alt + t

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

```
$ sudo apt-get install git
```

3) Git 명령어

분류	명령어	설명
<새로운 저장소 생성>	$ git init	.git 하위 디렉토리 생성 (폴더 생성 후 그 안에서 명령 실행 -> 새로운 git 저장소 생성)
<저장소 복제/다운로드 (clone)>	$ git clone <https:..URL>	기존 소스 코드 다운로드/복제

	$ git clone /로컬/저장소/경로	로컬 저장소 복제
	$ git clone 사용자명@호스트:/원격/저장소/경로	원격 서버 저장소 복제
<추가 및 확정 (commit)>	$ git add <파일명> $git add *	커밋에 단일 파일의 변경사항을 포함(인덱스에 추가된 상태)
	$ git add -A	커밋에 파일의 변경사항을 한번에 모두 포함
	$ git commit -m "커밋 메시지"	커밋 생성(실제 변경사항 확정)
	$ git status	파일 상태 확인
<가지(branch)치기 작업>	$ git branch	브랜치 목록
	$ git branch <브랜치이름>	새 브랜치 생성 및 이동
	$ git checkout -b <브랜치이름>	브랜치 생성 및 이동
	$ git checkout master	Master branch로 되돌아옴
	$ git branch -d <브랜치이름>	브랜치 삭제
	$ git push orgin <브랜치이름>	만든 브랜치를 원격 서버에 전송
	$ git push -u < remote > < 브랜치이름>	새 브랜치를 원격 저장소로 push
	$ git pull < remote > <브랜치이름>	원격에 저장된 git 프로젝트의 현재 상태를 다운받고 현재 위치한 브랜치로 병합
<변경 사항 발행 (push)>	$ git push origin master	변경사항 원격 서버에 업로드
	$ git push < remote > <브랜치이름>	커밋을 원격 서버에 업로드

	$ git push -u < remote > < 브랜치이름>	커밋을 원격 서버에 업로드
	$ git remote add origin <등록된 원격 서버 주소>	클라우드 주소 등록 및 발행(git에게 새로운 원격 서버 주소 알림)
	$ git remote remove <등록된 클라우드 주소>	클라우드 주소 삭제
<갱신 및 병합 (merge)>	$ git pull	원격 저장소의 변경 내용이 현재 디렉토리에 가져와지고 병합됨
	$ git merge <다른 브랜치이름>	현재 브랜치에 다른 브랜치의 수정사항 병합
	$ git add <파일명>	각 파일을 병합할 수 있음
	$ git diff <브랜치이름><다른 브랜치이름>	변경 내용 merge 전에 바뀐 내용을 비교할 수 있음
<태그tag작업>	$ git log	현재 위치한 브랜치 커밋 내용 확인 및 식별자 부여
<로컬 변경사항 return 작업>	$ git checkout -- <파일명>	로컬의 변경사항을 변경 전으로 되돌림
	$ git fetch origin	원격에 저장된 git프로젝트의 현 상태를 다운로드

- 브랜치(branch)란 간단히 하나의 프로젝트를 여러 갈래로 나누어서 관리할 수 있도록 하는 기능이다. 각각의 독립된 Branch에서 마음대로 소스코드를 변경하여 작업 한 후 원래 버전과 비교하여 또 하나의 새로운 버전을 만들어 낼 수 있다.
- 커밋(commit)이란 프로젝트의 현재 상태를 나타내는 체크포인트 또는 스냅샷이라 할 수 있다. 간단히 현재 버전의 코드를 커밋에 저장한다고 생각하면 된다.

- Push : git 저장소에 변경점 올리기 / Pull : git 저장소에서 변경점 가져오기
  사용법 : $ cd 'git push할 폴더 경로' -> $ git init //최초 한번만 -> $ git remote add <저장소 이름> <저장소 경로> //최초 한번만 -> ~
- 레포지토리(Repository)란 Git에서 코드를 저장하는 공간 또는 Git으로 관리하는 프로젝트 저장소이다. 레포지토리는 자신의 컴퓨터 작업공간(local)에 위치한 로컬 Git 레포지토리와 Github 등의 원격(remote) 공간에 위치한 원격 레포지토리로 구분한다.
- Submodule : Git의 레포지토리 하위에 다른 저장소를 관리하기 위한 도구이다. 이때 상위 레포지토리를 슈퍼 프로젝트(superproject), 하위 레포지토리를 서브 모듈(submodule)이라고 부른다.

  Submodule이 있는 저장소 clone(내 로컬 컴퓨터로 복사해오기) 하는법 : $ git clone <git 저장소 주소> -> $ cd <clone 해온 폴더 이름> -> $ submodule init -> $ git submodule update

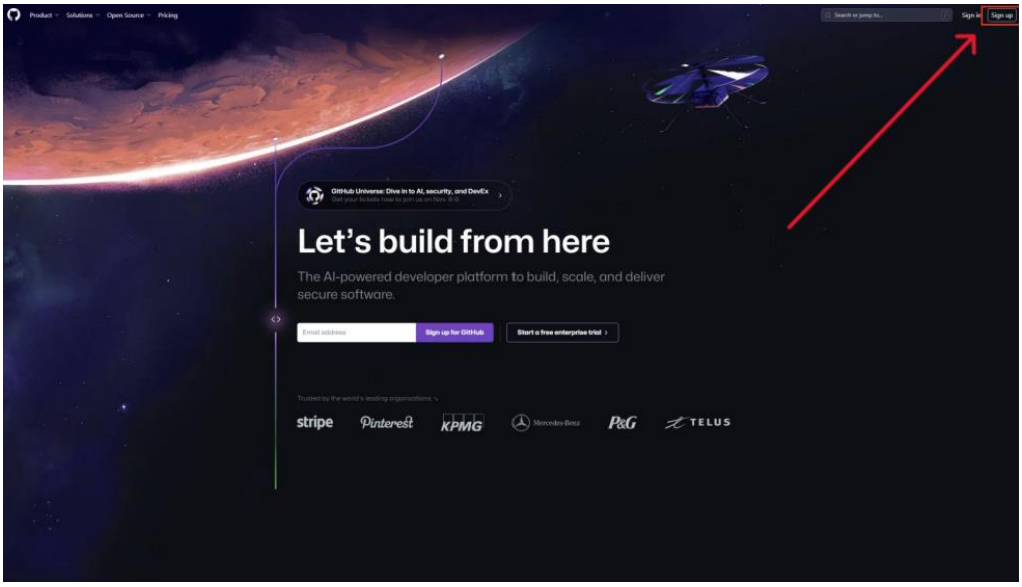
  Submodule을 등록하는 방법 : $git submodule add <git 저장소 주소>

4) Git 토큰 생성

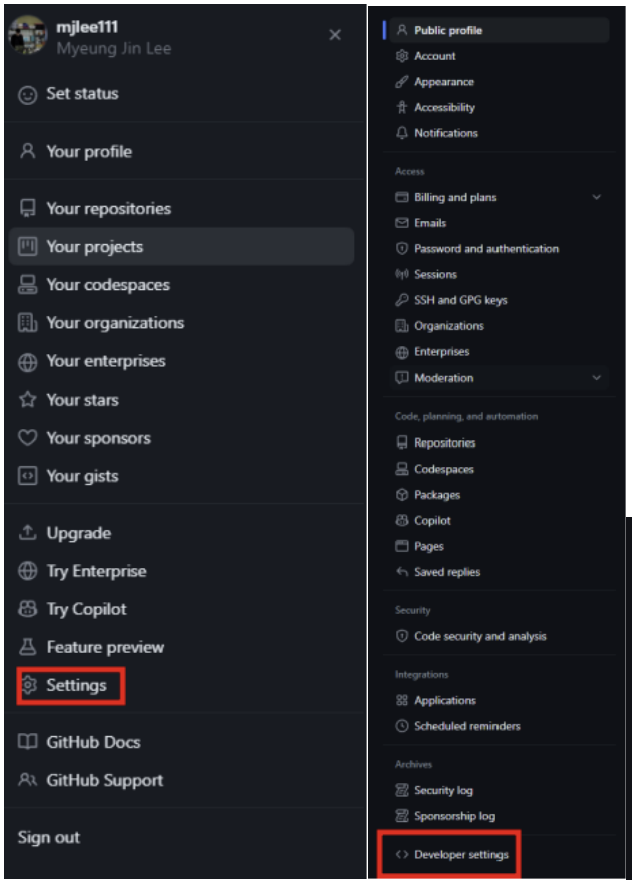
Git 토큰이란 간단히 내 계정의 아이디랑 비밀번호를 대신하는 문자열이다.

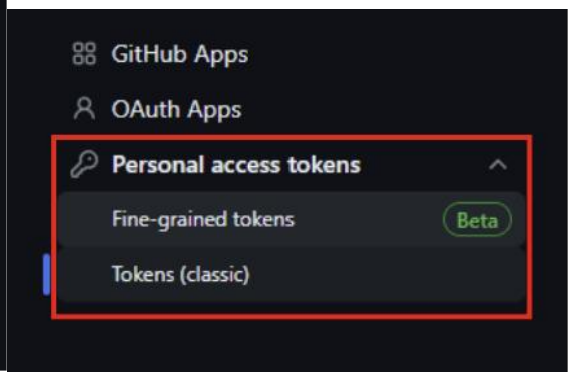
이 토큰을 생성하는 이유는 터미널에서 password처럼 사용할 수 있기 때문이다.

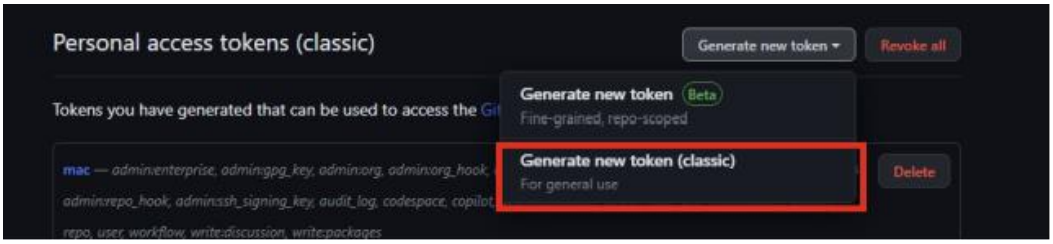
먼저 Git 계정을 생성한다. (https://www.github.com/)



이제 토큰을 생성. 아래 사진들/빨간색을 따라가면 됨.

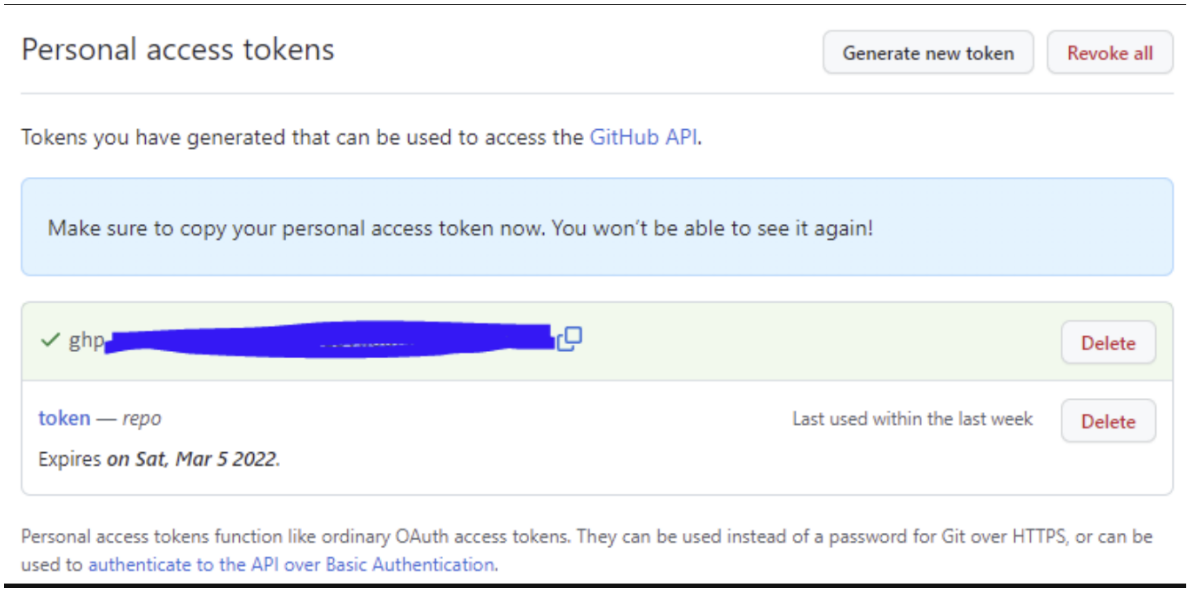






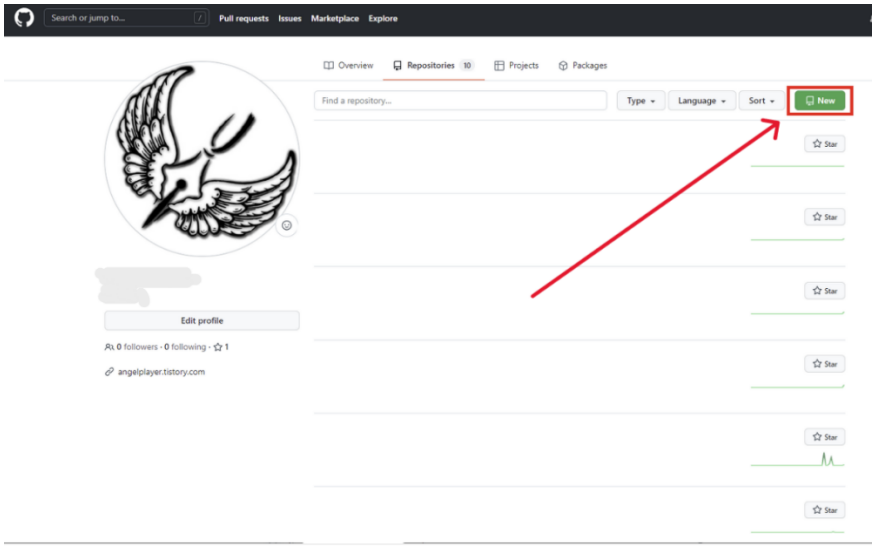
빨간 색 부분에서 토큰명을 입력하고 토큰 권한부분에서 원하는 권한들을 선택한다(기본적으로 레포지토리를 사용하기에 repo부분은 선택). -> 토큰 생성.

이렇게 토큰을 생성하게 되면 고유번호를 알려주는 화면이 출력되는데 여기서 토큰 고유번호(비밀번호)는 이때가 아니면 다시 볼 수 없기 때문에 복사 후 다른 곳에 잘 저장해 놓아야 한다.

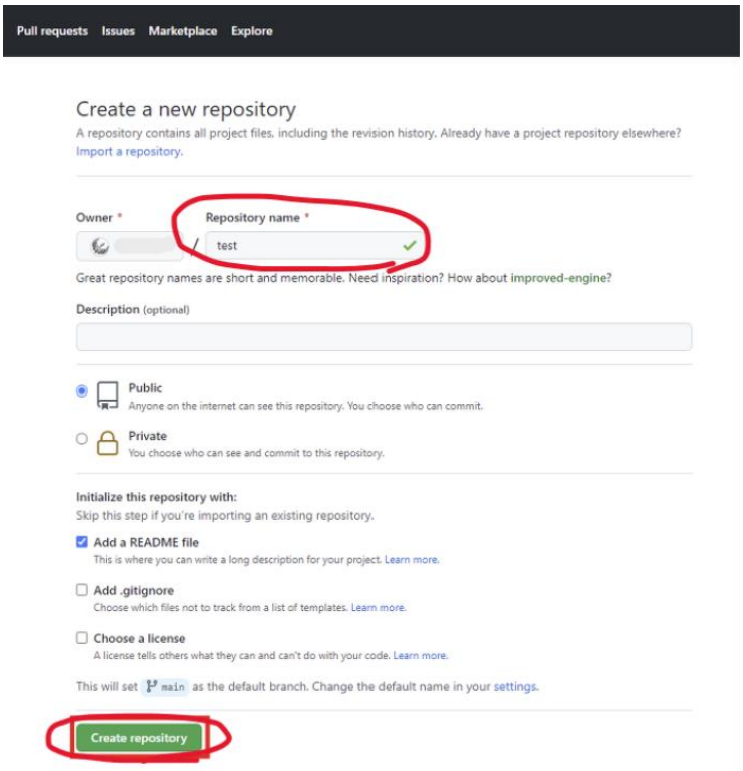


5) 레포지토리(Repository) 생성 및 Push

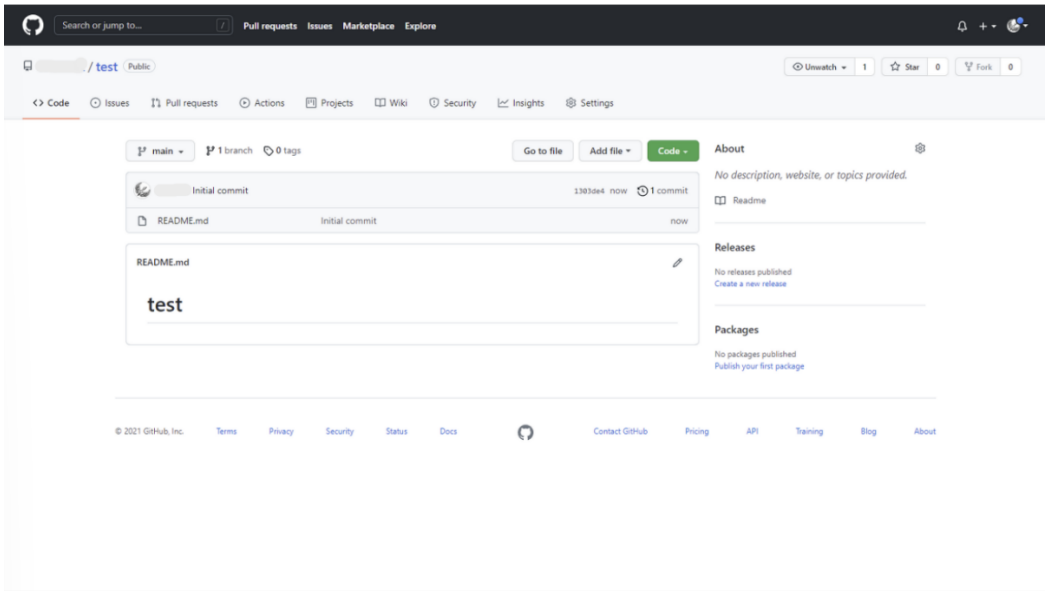
gitbuh접속 후 Repositories로 이동하면 [New] 클릭.



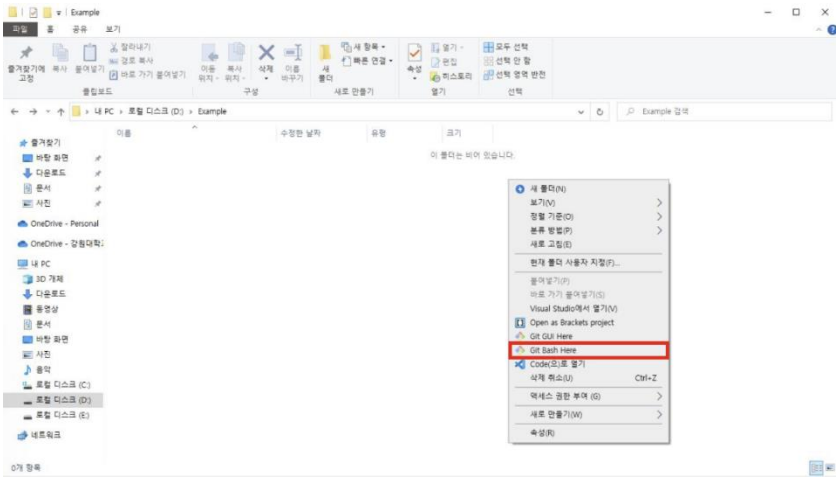
Repository 생성 화면에서 Name을 설정 후 [Create repository] 클릭.



그러면 Repository가 일단 생성이 완료된다.



이제 내 PC에 github repository를 설정하고 Push해보자. 먼저 git 저장소를 내려받을 위치(파일)를 생성한다(=mkdir ~). 그리고 이 파일을 git 원격 저장소로 지정해야 하기 때문에 cd 파일명 으로 지입해 git init으로 .git파일을 생성해주거나 파일을 우클릭 후 git bash here을 선택해 생성한다.

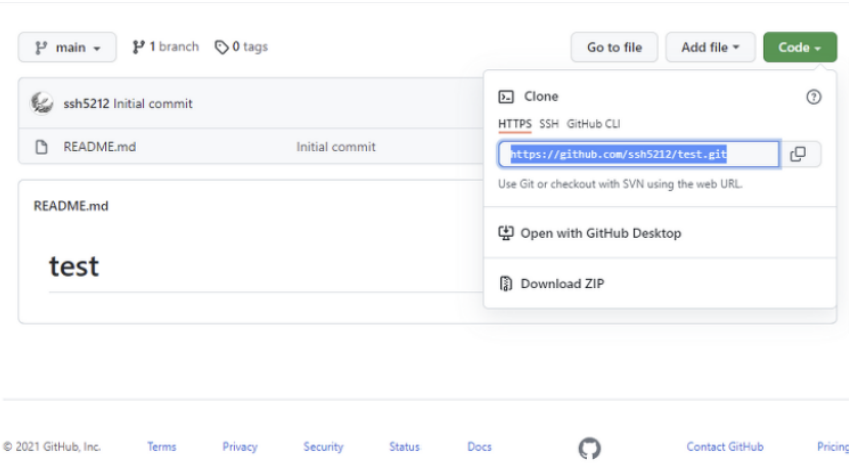


여기에 $ git config –global user.name [github name] $ git config –global user.email [github email] 가입 이름과 이메일 두 라인을 입력한다.

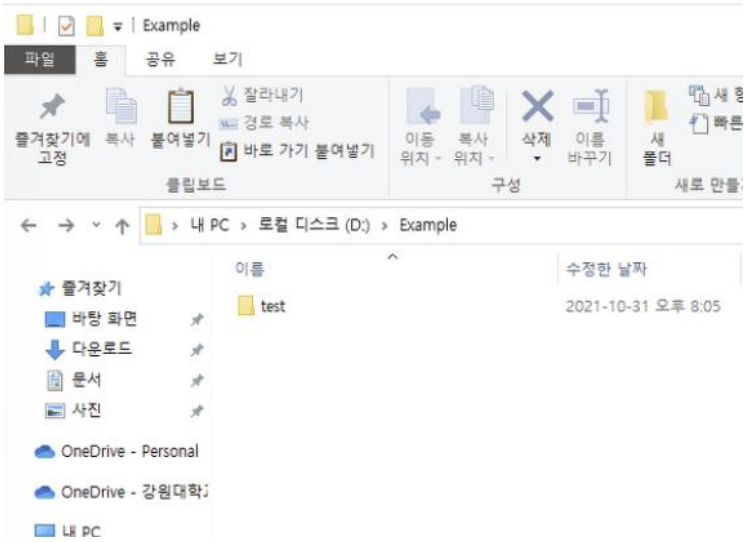
참고로 .git파일은 생성한 파일 내분에 있을 것이지만 처음엔 보이지 않을 것이다. 이를 보기 위해선 파일의 설정에서 숨긴 파일을 보이게 해야한다.

설정이 완료되면 github repository로 이동하여 [code] 클릭 후 나타나는 링크를

복사한다.



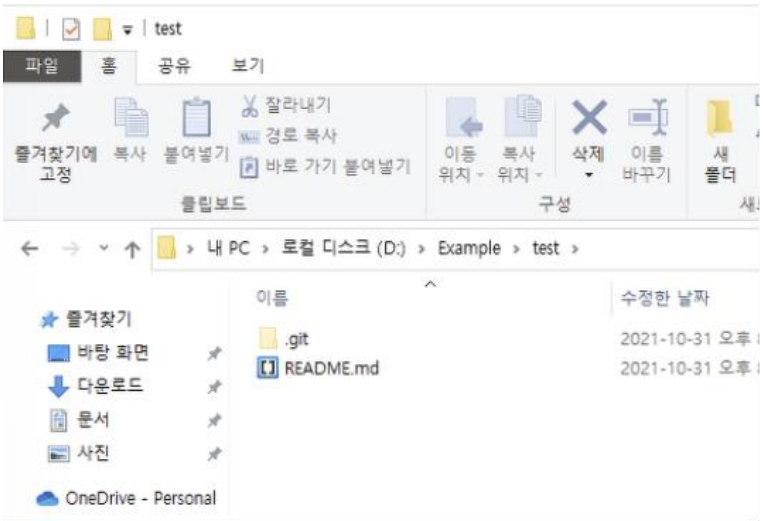
아까 커맨드 창에 다음 라인을 입력한다. $ git remote add <저장소 별칭(ex:origin)> <저장소 주소(ex:위 복사한 링크)> 이 명령어를 통해 생성한 파일과 git 사이트 저장소를 연결한 것이다.



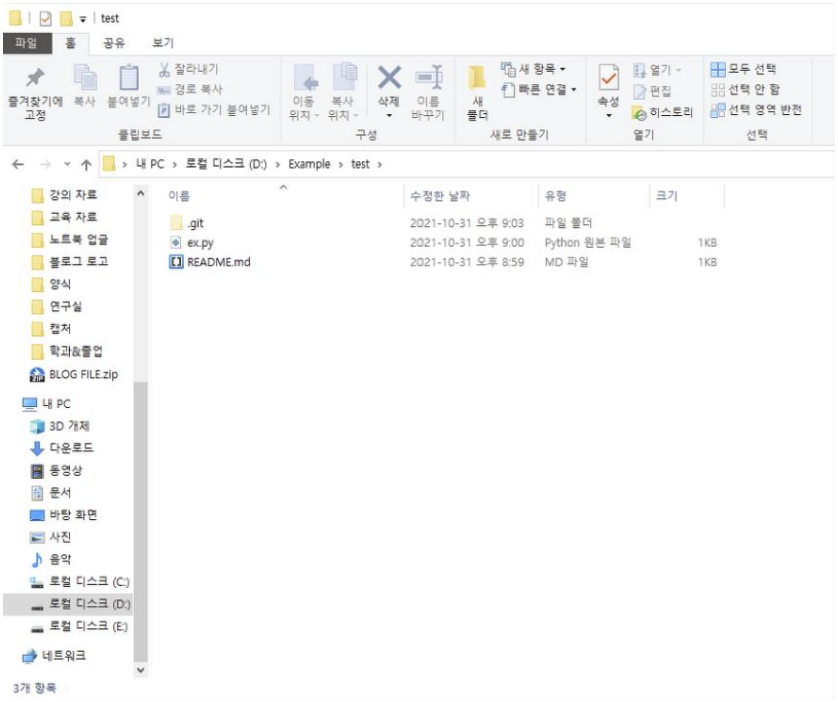
폴더 내부에 git폴더가 생성되었고, git remote -v를 입력했을 때 다음 사진과 비슷하게 나온다면 저장소와 제대로 연결 및 작동된 것이다.

그리고 저장소와 연결된 파일(사진에선 test파일) 내부에 Push하고 싶은 업로드

파일을 넣어준다.(아래 사진에선 README.md 파일을 업로드)



그리고 터미널창에 git status명령어를 입력하면 새로 넣어지거나 수정된 파일이 빨간 글씨로 출력되는데 자기가 업로드하고 싶은 파일이 빨간 글씨의 파일명과 같다면 제대로 넣어지거나 수정된 것이다.



업로드 과정 명령어를 보자면, git add <업로드 파일명> 혹은 git add .(파일에 있는 모든 변경사항 업로드)를 입력 후 git commit -m "커밋메시지" (업로드에 대한 보충설명)를 입력한다. 그리고 git push <저장소 별칭(ex:origin)> <브랜치명(ex:master)> 입력하면 아래 사진과 같이 출력된다.

```
sangs@ubuntu:~/PycharmProjects/test_project$ git push origin HEAD:main
Username for 'https://github.com': laoching
Password for 'https://laoching@github.com':
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (19/19), 2.49 KiB | 2.49 MiB/s, done.
Total 19 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/laoching/test_field.git
   84008db..e3876bf  HEAD -> main
```

여기서 Username을 입력하라고 하는데 이는 말 그대로 원하는 name을 입력하면 되지만 Password입력 부분에는 앞서 발급한 토큰 고유번호(비밀번호)를 복사하여 여기에 입력해야 한다. 그렇게 입력해서 제대로 작동한다면 git사이트에 업로드한 파일이 올라가 있는 것을 볼 수 있을 것이다.

참고로 유저들 간에 커밋메시지를 작성하는 7가지 규칙이 정해져 있는데 그 규칙은 다음 과 같다.

1. 제목과 본문을 빈 행으로 구분한다.
2. 제목은 50글자 이내로 제한한다.
3. 제목의 첫 글자는 대문자로 작성한다.
4. 제목 끝에는 마침표를 넣지 않는다.
5. 제목은 명령문으로 사용하며 과거형을 사용하지 않는다.
6. 본문의 각 행은 72글자 내로 제한한다.
7. 어떻게 보다는 무엇과 왜를 설명한다.

커밋메시지 구조는 다음과 같고

```
// Header, Body, Footer는 빈 행으로 구분한다.
타입(스코프): 주제(제목) // Header(헤더)

본문 // Body(바디)

바닥글 // Footer
```

여기서 타입(스코프) 종류는 다음과 같다.

타입 이름	내용
feat	새로운 기능에 대한 커밋
fix	버그 수정에 대한 커밋
build	빌드 관련 파일 수정 / 모듈 설치 또는 삭제에 대한 커밋
chore	그 외 자잘한 수정에 대한 커밋
ci	ci 관련 설정 수정에 대한 커밋
docs	문서 수정에 대한 커밋
style	코드 스타일 혹은 포맷 등에 관한 커밋
refactor	코드 리팩토링에 대한 커밋
test	테스트 코드 수정에 대한 커밋
perf	성능 개선에 대한 커밋

바닥글에 대한 종류도 예시로 있는데 close, closes, closed, fix, fixes, fixed, resolve, resolves, resolved 등이 있다. 이 바닥글에서는 #<이슈번호>를 추가하여 다른 코드, 이슈와 연결 및 보충설명을 해줄 수 있다.

커밋메시지 작성 예시는 다음과 같다.

```
git commit -m "fix: Safari에서 모달을 띄웠을 때 스크롤 이슈 수정

모바일 사파리에서 Carousel 모달을 띄웠을 때,
모달 밖의 상하 스크롤이 움직이는 이슈 수정.

resolves: #1137
```

5. 드라이버

1) 우분투 업데이트 및 terminator설치

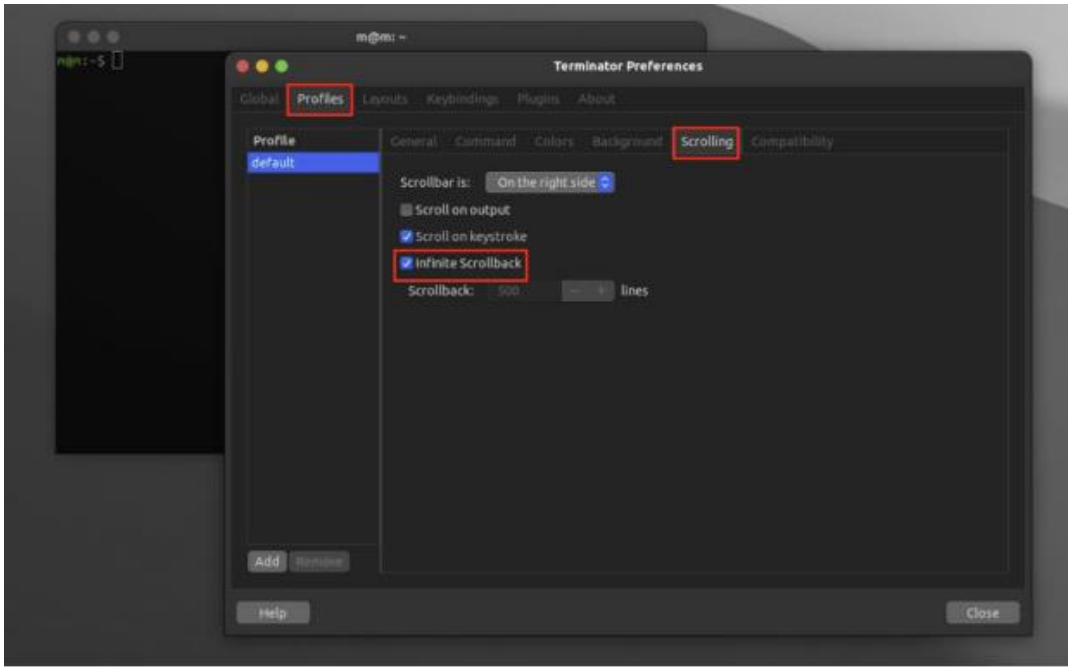
```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

```
$ sudo apt-get install terminator
```

중간에 y/n 나오면 y 입력 후 엔터. Terminator 설치 후 터미널 창 닫은 후 다시 실행.

2) Terminator 설정



터미널창 우클릭 -> Preferences클릭

-> Profiles-scrolling -> Infinite scrollback활성화

하는 이유 : 터미널 줄바꿈 횟수가 무제한으로 설정되어 디버깅 시 용이함.

6. OpenCV

1) OpenCV란

OpenCV란 Open Source Computer Vision의 약자로 영상 처리에 사용할 수 있는 오픈 소스 라이브러리이다. 이미지 프로세싱, 컴퓨터 비전 및 머신러닝 알고리즘에 대한 다양한 함수와 라이브러리를 제공한다. OpenCV는 이미지나 동영상에서 물체검출, 추적, 특징추출, 패턴인식, 얼굴인식 등의 작업을 수행할 수 있으며 이 외에도 이미지 필터링, 색 공간 변환, 이미지 모핑, 카메라 보정 등 다양한 기능을 제공한다. OpenCV는 무료로 제공되며, 비즈니스 및 개인 프로젝트 모두에서 사용 가능하다. 또한, 여러 플랫폼에서 동작하므로 모바일, 웹, 데스크탑 등 다양한 환경에서 사용할 수 있다.

2) OpenCV 4.5.2설치

```
$ sudo apt-get install git
```

```
sudo apt-get install git
```

```
cd
```

```
git clone https://github.com/mjlee111/OpenCV_auto_installer.git
```

```
cd OpenCV_auto_installer
```

```
sudo chmod +x opencv_installer.sh
```

```
./opencv_installer.sh
```

한 줄씩 입력&엔터 후 모두 입력해서 정상 작동하면 running 상태가 된다.

Running이 끝난 후 OpenCV버전인 4.5.2를 입력.

(= https://github.com/mjlee111/OpenCV_auto_installer로 가서 README 파일 읽으며 설치.)

3) CV_bridge config cleaner 실행

```
sudo bash CVBRIDGE.sh
```

(= https://github.com/mjlee111/OpenCV_auto_installer로 가서 README 파일 읽으며 설치.)

4) .bashrc 환경변수 파일 수정

$ sudo gedit ~/.bashrc -> 사진 가장 아래쪽에 왼쪽 텍스트 입력 -> 저장 후 창 닫고($ source ~/.bashrc 에러 무시) -> $ cm $sb $ sudo reboot

```
alias gb='cd && gedit .bashrc'
alias eb='nano ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gb='gedit ~/.bashrc'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
alias cb='source devel/setup.bash'
alias depinstall='rosdep update && rosdep install --from-paths . --ignore-src -r -y'
alias rn='sudo systemctl restart NetworkManager'
alias symlink='sudo udevadm control --reload-rules && sudo udevadm trigger'
alias killg='killall -9 gzserver && killall -9 gzclient && killall -9 rosmaster'
```

```
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
```

(Bash는 Bounce Again Shell의 약자이며 리눅스에서 사용자정의 명령어를 만들어 사용할 때 사용되는 파일이다. 숨긴 파일로 존재하기에 ls로 확인할 수 없고 ls -a 명령어로 확인 가능하다. bashrc는 사용자가 원하는 명령어를 만들 수 있는 파일이다.)

5) 설치 확인

```
$ roscore
```

에러 안뜨고 잘 나오는지 확인

```
$ sudo pkg-config – modversion opencv4
```

4.5.2 출력 확인